Aalto University
School of Science
Degree Programme in Computer, Communication and Information Sciences

Duc Hung Luu

Deploying building information modeling software

on Desktop as a Service platform

Master's Thesis
Espoo, March 09, 2017

Supervisor: Professor Antti Ylä-Jääski, Aalto University
Advisors: Paula Hellemaa, M.Sc. (Tech.)
Mirko Gatto, Trimble Solutions Corporation

Aalto University
School of Science
Degree Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author:	Duc Hung Luu		
Title:	Deploying building information modeling software on Desktop as a Service platform		
Date:	March 09, 2017	**Pages:**	68
Major:	Mobile Computing, Services and Security	**Code:**	T-110
Supervisor:	Professor Antti Ylä-Jääski		
Advisors:	Paula Hellemaa, M.Sc. (Tech.) Mirko Gatto, Trimble Solutions Corporation		

Desktop as a Service (DaaS) is a novel cloud computing service that provides cloud-based virtual desktops on-demand to end users. The major advantage of DaaS is the capability to quickly deliver expeditious control of a full desktop environment to end users from various device platforms such as Android, iOS, MacOS or Web access from anywhere and at any time.

This master thesis is a proof of concept to demonstrate the practicability to deploy the case company's graphics-intensive building information modeling software, Tekla Structures on Amazon Web Services' DaaS solution, named Amazon WorkSpaces. We investigated the whole deployment process of the software to the Amazon WorkSpaces. After clarifying the deployment process, we developed the working prototype consisting of different Amazon Web Services to automate the process. Furthermore, we implemented operational test cases for the prototype and for the Tekla Structures running on Amazon WorkSpaces to determine the feasibility of using this novel cloud service for the production purpose in the case company.

In summary, Amazon WorkSpaces is a highly anticipated DaaS solution that can simplify the desktop and software delivery process to the case company's customers. The prototype developed in the thesis can automate the deployment process and launch new Amazon WorkSpaces to a sufficient extent. Moreover, the evaluation shows that the prototype can handle its automation tasks correctly based on the proposed architectural design and the Amazon WorkSpaces with Graphics hardware configuration are capable of operating Tekla Structures impeccably as in physical Windows desktops.

Keywords:	Amazon WorkSpaces, Desktop as a Service, virtual cloud desktop, cloud computing, Tekla Structures, graphics-intensive software, continuous deployment
Language:	English

Acknowledgements

This thesis was written for Trimble Solutions Corporation. I wish to thanks Trimble for an interesting thesis topic and resources to work on this thesis.

I want to thank my supervisor, Professor Antti Ylä-Jääski for the valuable feedback and suggestions that ushered my thesis to the right path. I am profoundly grateful to have Paula Hellemaa and Mirko Gatto as my thesis advisers. Thank you for your dedicated guidance during the whole process of this thesis work.

I would like to express my gratitude to my manager Jari Patanen and the entire Software Deployment team for supporting my thesis work and reminding me of lounasaika.

My thanks go to Software Architecture, SOLD teams and other colleagues at Trimble Solutions Finland whom I have had thought-provoking discussions that keep me inspired and motivated.

Thank you my family for being always by my side!

Finland, March 09, 2017

Duc Hung Luu

Abbreviations and Acronyms

Amazon EC2	Amazon Elastic Compute Cloud
Amazon S3	Amazon Simple Storage Service
Amazon WAM	Amazon WorkSpaces Application Manager
AD	Active Directory
API	Application programming interfaces
AWS	Amazon Web Services
CD	Continuous Delivery
CI	Continuous Integration
CSP	Cloud service providers
CSU	Cloud service users
DaaS	Desktop as a Service
DAaaS	Desktop Application as a Service
GPU	Graphics processing unit
PCoIP	PC-over-IP
JSON	JavaScript Object Notation
VCS	Version Control System
VDI	Virtual Desktop Infrastructure
REST	Representational State Transfer

Contents

List of Figures

Chapter 1

Introduction

Cloud computing refers to an evolving paradigm that is shifting client/server systems to hosted services. Over the last decade, cloud computing has advanced from a precarious concept to a mature and prevalent technology that provides flexible, on-demand information technology (IT) systems over the Internet. Cloud computing power comes from physical data centers located in one or many different places that are pooled and virtualized to form a multi-tenant network architecture.[27] Furthermore, cloud computing is cost effective since it removes the need of on-premises IT infrastructure and operations. There are various cloud computing service models such as Software as a Service, Platform as a Service, Infrastructure as a Service, Compute as a Service and many more. Nevertheless, this thesis focuses particularly on Desktop as a Service (DaaS) model.

DaaS is a relatively novel cloud computing service that offers full cloud-based virtual desktop experience for end users on various platforms, e.g., Android, iOS, Windows, MacOS and web access. Today, with the substantial growth of portable devices and broadband Internet connection, the demands to get instant access to data, applications and desktops on any devices anywhere and at any time has become imperative. This trend demands enterprise IT departments to conduct a more secure and reliable access remotely to workplace desktops.

With DaaS, end users only need a simple device that can run a DaaS client software to connect and control their virtual desktops. The concept of DaaS is highly practical for users that require accessing to full desktops without taking care of the tedious configuration and hardware maintenance. Consequently, DaaS is exposed as an ideal solution to fulfill these demands.

This thesis work stemmed from the need to deliver customers an enterprise DaaS solution that can operate the case company software without

compromising security and performance. Particularly, the thesis aims to deploy and evaluate the feasibility of implementing Tekla Structures, the case company software, on Amazon WorkSpaces, a DaaS solution in Amazon Web Services.

1.1 Problem statement

Tekla Structures is a Building Information Modeling (BIM) software of the case company. As the nature of BIM software that usually involves drawing and rendering multiplex 3D models, Tekla Structures requires intensive graphics computation to function smoothly. Generally, customers work with Tekla Structures using high performance workstation desktops and laptops. However, as the cloud computing becomes more ubiquitous, and demands to get instant access to the software regardless of devices and location are arising. It is essential for the case company to provide a solution to meet these demands. DaaS stands out to be a good candidate to provide a reliable control of virtual desktop machines that are capable of running graphics-intensive software such as Tekla Structures.

Amazon WorkSpaces is a newly introduced DaaS solution within Amazon Web Services (AWS) ecosystem. As the case company has been adopting AWS extensively, taking Amazon WorkSpaces into use would not only help to develop a more diverse Trimble Solutions Corporation product offering in AWS but also fulfill the high-quality cloud desktops for customers. Additionally, Amazon WorkSpaces offers high-end GPU-powered virtual desktops that meet the Tekla Structures requirement specifications.

On the other hand, the case company product development generates several Tekla Structures versions daily from the in-house continuous delivery system. It is important to automate the deployment process of packaging Tekla Structures to Amazon Workspaces as wholly as possible. The automation process would help to deliver the software to end users in a timely manner and minimize repetitive, error-prone tasks. Currently, the case company is providing Citrix XenApp and XenDesktop as DaaS solutions for delivering Tekla Structures software to end users. By experimenting Amazon WorkSpaces, we can compare and assess equitably the benefits of each DaaS solution.

1.2 Scope of the Thesis

The main objective of this thesis is to deploy Tekla Structures on Amazon WorkSpaces in Amazon Web Services and automate this whole deployment process. The thesis aims to produce a proof of concept on how feasible Tekla Structures can be used on Amazon WorkSpaces.

In order to attain the objective, we investigate the technologies that can be utilized for automating the deployment process. In essence, a multitude of AWS services and scripting languages are studied and developed for automating these tasks. We scrutinize the design of the Amazon WorkSpaces and its required packaging work-flow for virtualizing and validating Tekla Structures. Additionally, we investigate pre-requisite and add-on applications needed to manage and operate Tekla Structures, e.g., Tekla license server, model sharing tool and customized environments. Then we examine the practical tasks to automate in the Amazon WorkSpaces application packaging and launching operations. Ultimately, the automation prototype is implemented according to the study and the proposed design. Once the prototype is developed and Tekla Structures fully functions in Amazon WorkSpaces, we perform different tests to assess the feasibility of using Tekla Structures in Amazon WorkSpaces. Furthermore, other imperative factors that are essential for any enterprise DaaS solutions such as security and reliability are reviewed.

Our research questions have been identified to achieve the objectives, and are listed below:

RQ1: How Tekla Structures can be deployed in Amazon WorkSpaces? Will Amazon WorkSpaces be capable of running Tekla Structures?

This thesis deploys Tekla Structures in Amazon WorkSpaces in order to understand the behavior of the Amazon WorkSpaces service as a whole. The findings from the implementation can be used to create a concrete plan for developing the continuous deployment system for the production use in the case company. Additionally, this research question aims to discover any limitations occurred in the deployment process.

RQ2: To what extent can continuous deployment tasks be automated and how to implement these tasks in Amazon Web Services?

To answer this question, we study the architecture of Amazon WorkSpaces. We investigate how to deploy different Tekla Structures versions on Amazon WorkSpaces and to automate these deployment tasks. By carrying out different trial-and-error implementations, the thesis evaluates the most sensible approach to be used for Amazon WorkSpaces in Amazon Web Services.

The output of this thesis work is an automated continuous deployment system built on different AWS services. The system aims to pave the way for taking Amazon WorkSpaces into use as a part of the case company's continuous deployment process and a DaaS offering for customers.

1.3 Structure of the Thesis

The thesis is divided into nine chapters. A brief introduction to the thesis work and the thesis objectives are provided in this chapter. Subsequently, Chapter 2 presents the background of the core technologies studied in the thesis. In Chapter 3, the case company information, their software and the continuous delivery system are provided. Chapter 4 is devoted to analyzing the AWS ecosystem in order to understand the AWS services used in the implementation. In Chapter 5, the design for the deployment of Tekla Structures in AWS is proposed. The implementation and system prototype are developed and scrutinized in Chapter 6. We assess various aspects of the system in Chapter 7. Chapter 8 is dedicated for discussing the findings of the thesis and suggesting directions for future studies. Finally, Chapter 9 concludes the thesis and synthesizes the results of the implementation.

Chapter 2

Background

2.1 Virtual Desktop Infrastructure

Virtual Desktop Infrastructure (VDI) is a centralized server-based computing platform that hosts user desktops in a data center. VDI enables multiple user desktops to have an independent operating system in a virtual machine while still sharing underlying physical hardware resources [1, 10]. VMware ESX server is the pioneer in the VDI approach that provides a thin software layer to allocate efficiently hardware resources among virtual machines. For example, a VMware ESX server can host and distribute computing power among Microsoft Windows 7, Microsoft Windows Server and Linux virtual machines concurrently [39]. Thus, multiple users can work on their virtual machines without affecting each others' activities via the remote desktop protocol.

VDI brings an ease to IT management as it allows central management and full control on software being installed and used on the virtual desktops. Additionally, deploying new virtual desktops is significantly faster compared to the physical desktop deployment. Security enhancement is another advantage in VDI. The VDI image can be locked from external devices and the data is centrally stored in the database server. Hence, if any devices used to connect to virtual machines are lost or stolen, all the information is still safe and protected in the server. Nevertheless, one drawback of VDI technology is the high investment and complexity to equip the data center infrastructure capable of virtualization [10].

2.2 Cloud computing

A consensus of cloud computing defines its concept as a model that allows omnipresent and on-demand network access to a shared pool of elastic, pay-per-use computing resources such as applications, storage and servers. Moreover, these resources can be easily provisioned and released by cloud service providers (CSP). [27]

In order to be considered as cloud computing, a computing service needs to demonstrate the five following essential characteristics:

1. On-demand self-service: Enabling cloud service users (CSU) to provision computing capabilities with no intervention from CSP.

2. Rapid elasticity: The ability to appropriately scale inwards or outwards the service depending on the demand in any quantity at any time.

3. Resource pooling: A multi-tenant model consisting of different physical and virtual resources that is pooled and assigned dynamically irrespective of the real location.

4. Measured service: Capabilities to monitor, control and optimize pooled resources usage and reporting a transparent usage and associated costs for CSU and CSP.

5. Broad network access: Resources are made available with universal high bandwidth, low latency and stable network access that can be accessed by heterogeneous client platforms such as laptops, tablets and mobile phones.

2.2.1 Cloud deployment models

Cloud deployment models regulate who can employ cloud resources and how these resources are accessed and located [42]. Cloud deployment practices are carried out based on four varied but rigidly connected models, i.e., public, private, hybrid and community. Figure 2.1 illustrates the deployment models concepts and their relations.

Private cloud infrastructure is designed exclusively for a single organization consisting of multiple business units. The private cloud should handle a specific function for business success and locate on or off the organization premises [27]. Public cloud is a multi-tenant cloud infrastructure shared by different users who usually have nothing in common and hosted on the premises of the cloud provider. Community cloud is utilized by an association of organizations that have shared interests. The community cloud is

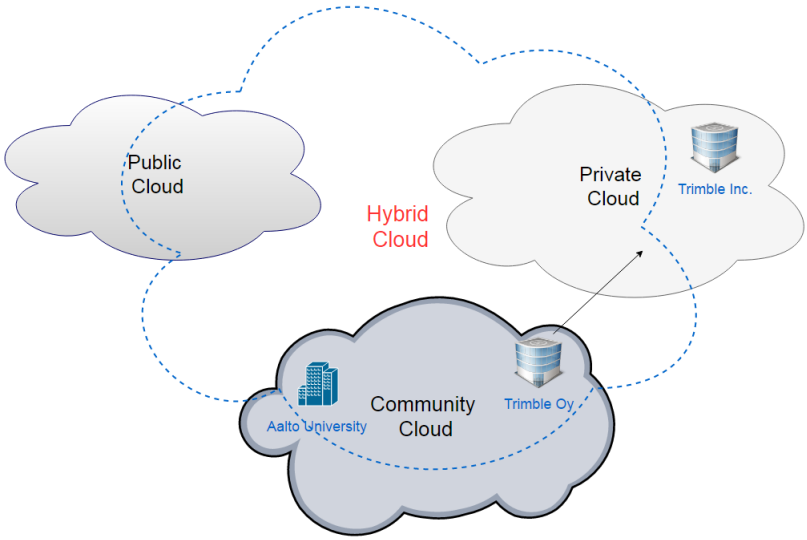


Figure 2.1: Four cloud deployment models

maintained by these organizations and/or a third vendor. As can be seen in the Figure 2.1, hybrid cloud is a mixture of two or more unique cloud deployment models (private, public, community) that remain as independent entities but are bound together as single cloud under a strong management framework [40].

Each cloud deployment model offers its advantages as well as challenges. There is no silver bullet model that can resolve all demands. Therefore, it depends on the organization to choose what cloud models can fulfill their needs and requirements.

2.2.2 Cloud computing architecture and services

The conventional cloud computing is divided into three main layers, i.e., application, platform and infrastructure. Each of these layers can be configured and managed by CSU. Alternatively, they can be supplied by CSP to form three major corresponding delivery models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Infrastructure layer can be seen as the fundamental foundation of the cloud computing. In the past, mainframe computers and rack servers operated monolithic applications. Today, virtualization[1] optimizes the hardware resources by allowing one server to run as multiple servers. As a result, the

[1]Virtualization: the abstraction of physical computing resources such as processors, storage devices, GPU cards and RAM.

hardware resources can be pushed to their full capacity and serve different utilities instantaneously [17]. IaaS is a way to deliver cloud computing infrastructure (server, storage and network) as an on-demand service. Thus, CSU can easily obtain ready-to-run IaaS from the cloud provider instead of equipping and managing the infrastructure of their own.

Platform layer provides CSU a computing platform to create, compile and test applications easily without the complexity of configuring the run-time, middleware, operating system and managing the underlying infrastructure. PaaS is most applicable when CSU wants to automate testing and deploy a product in a rapid and iterative software development.

Application layer is designed to distribute cloud native software to CSU. Instead of running on-premises, SaaS is often centrally managed and hosted by third-party vendor and the software is delivered to CSU over the Internet as a service on demand or pay-as-you-go model. [17]

As this thesis focuses on Desktop as a Service model, we devote the next section to discuss about this emerging model in more detail.

2.3 Desktop as a Service

2.3.1 Definition of Desktop as a Service

Desktop as a Service (DaaS) is a cloud service that offers on-demand virtual desktops running on the cloud to remote users over the Internet. The back-end of DaaS is a VDI hosted and maintained by the CSP. In essence, DaaS is the VDI in the cloud that delivers cloud-based virtual desktops. DaaS operates on a multi-tenant architecture and is provided as a subscription or pay-per-use service. The CSP supervises virtualization, server, security, data storage and backup which means that the complexity of VDI is concealed from the CSU. As a result, DaaS CSU can get all the advantages that VDI provides [29, 35] while not needing to concern about the underlying system.

Figure 2.2 adapted from Grossman L. [13] illustrates the cloud technology segments that are covered by the CSP and for what the CSU has to take responsibility in each cloud service models. Additionally, the figure shows where DaaS stands in the cloud service models. DaaS is frequently regarded as the graphical user interface based IaaS. The model is akin to IaaS since the CSU can fully utilize the infrastructure layer. However, one difference is that IaaS provides command line interface instead. Moreover, DaaS gives the CSU possibility to customize their platform layer by using configurations provided by the CSP and/or their own preferences. On the other hand, when comparing DaaS with SaaS where only Web applications are utilized,

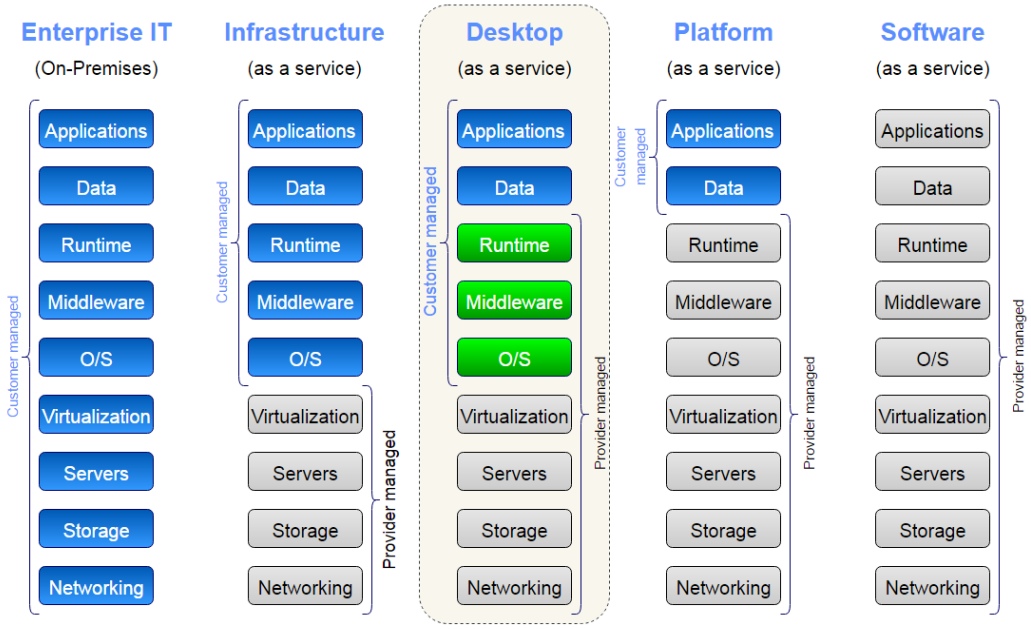


Figure 2.2: DaaS and other cloud service models in the cloud segments

all types of software applications can be used in DaaS as in a full desktop [21]. DaaS and desktop-applications as a service together build up a wider cloud model called Workspace as a Service.

2.3.2 Service Architecture for Desktop as a Service

This section discusses the standard functions and their interactions with the CSU in the DaaS service architecture. According to the International Telecommunication Union [18], there are four major service functions in a DaaS architecture as followed:

1. Connection brokering (CB): a software application can work as a connection broker that connects the CSU to an available virtual desktop. The connection broker application executes the user authentication and license verification between the CSU and CSU's software. The application also manages the virtual machine (VM) for user assignment and monitors the level of activity for the given VM. Additionally, the connection broker application can coordinate the deploying protocols between servers and users.

2. Resource pooling: A resource pool has to supervise high-capacity soft-

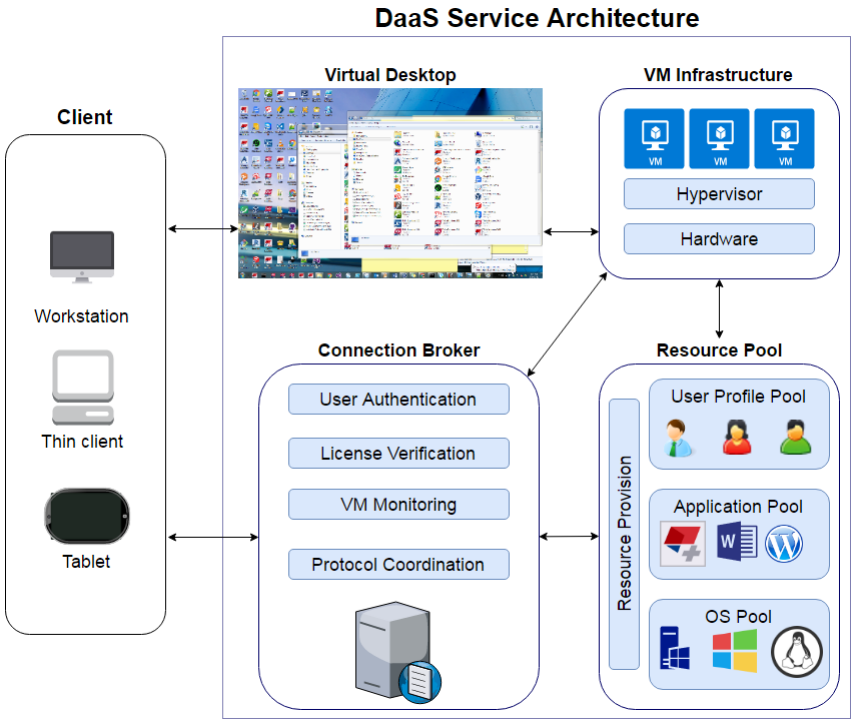


Figure 2.3: DaaS service architecture and the interactions with the client

ware resources such as OS, applications and user profiles[2] in order to ensure on-demand DaaS services. These software resources are streamed and run on a dedicated VM.

3. Virtual machine infrastructure: This function aims to create VMs and administer hardware resources. A hypervisor in the virtualized desktop server is applied to deploy dynamically the hardware resources to a higher level of software. As a result, the VM infrastructure produces virtual desktops from VMs that are used by the CSU.

4. Virtual desktop delivery: The main role of this function is to encapsulate and transfer the information system environment to a remote client device over the network. The virtual desktop delivery protocol is utilized to arrange communication channels to send and receive all the interaction information between servers and CSU's terminals in DaaS sessions.

[2]User profile: the individual information related to hardware configuration, operating environment, current OS and assigned applications

These four functions and their components are illustrated in Figure 2.3. Additionally, the figure presents the workflow when a client interacts with the DaaS service.

An end user only needs the simple client to connect and work with their virtual machine. Firstly, when a client requests an access to the CB via a security protocol such as SSH or TLS, the CB authenticates the client ID and associated password. If the verification succeeds, the CB checks for the client profile from the User Profile Pool. The Resource Pool also supports the CB to find the optimal VM based on the client's hardware configuration and software required. If no VM is feasible, the CB instructs the VM infrastructure to create new VM aligned with the client's configurations. After a VM is created, the CB applies the client profile to this VM, such as the selected OS and applications to formulate a virtual desktop. The VM infrastructure generates a connection and dispatches the connection information to the CB. The client uses this enquired information from the CB to connect to their virtual desktop. The client interacts with the virtual desktop via virtualized desktop delivery protocol, e.g., PC-over-IP. When the communication between the client and the virtual desktop stops, the CB is informed. The CB updates the most recent client profile to the user profile pool as well as returns the virtual desktop and its state to the VM infrastructure [18].

2.3.3 Desktop as a Service benefits and limitations

Cloud computing in general and DaaS specifically have brought many benefits for different parties utilizing the service.

For organizations, DaaS can deliver higher availability in comparison to physical on-premises workstations as virtual desktops are centrally managed in data centers and the workload can be spread across many facilities where they are constantly monitored for optimal service up-time. Data loss due to hardware malfunction or disaster occurrence can be prevented and business can continue with a minimal effect since CSU can easily make important data backed up and stored in multiple sites with DaaS. Besides that, the initial IT infrastructure investment can be eliminated as the IT operation is shifted from the upfront capital expense to the operational, pay-per-use expenses [19]. Furthermore, organizations can quickly scale and expand their cloud infrastructure as needed and only pay for the cloud services used without building a whole data center from the ground up.

From the IT administrator perspective, the IT management is simplified as all the virtual desktops can be supervised through a centralized infrastructure management system. The centralized control and efficient management of resources in DaaS also allows faster software and OS's provisioning time

at lower cost thus enhancing the service level agreements [35]. Additionally, administrative tasks would require less IT expertise at the organizational level since the complex infrastructure is handled by the CSP.

DaaS end users can be more productive and flexible. The end users can get access to their data and applications on any device irrespective of place and time. It means that DaaS supports the users with the highly mobile desktop-powered service and business can be accomplished anywhere and at any time. For instance, in the construction industry, general contractors in the office, architects and engineers in the construction sites around the globe can collaborate and work on a same construction project simultaneously. Hence, conflicts can be eliminated in early decision making stage to ensure that all teams are working towards the common goal and having the mutual understanding of the project [5]. Moreover, with virtual desktops delivered to different devices such as tablets or smart-phones in DaaS, construction engineers can monitor on-site how the implementation is carried out in comparison to the model created in the desktop Building Information Modelling software and update the model at the same time.

However, there are few limitations existed in DaaS. Due to the nature of the cloud computing that requires network connection to communicate with data centers, DaaS would need to be always accessed to the Internet for the service to function properly. Security is also a big concern when CSU start to use DaaS. IT administrators may not know exactly where the data is physically stored and the organizations' security policies might be dissimilar with what DaaS providers have to offer. From technical side, hypervisors in VM infrastructure usually enclose a privileged domain that is capable of accessing the VM's active memory. If a hypervisor was compromised, the memory content, virtual network traffic and other communication data that it controls could be exploited [11, 23]. As DaaS is a relatively new service and computational paradigm is different from the traditional standalone desktop, virtual desktop hardware specifications that DaaS providers offer is fairly restricted, particularly in graphics processing capability.

2.3.4 Desktop as a Service providers

Regardless of its late start, DaaS is gradually replacing VDI and thriving as a notable cloud service delivery model. Today, the DaaS market is going through a rapid growth as indicated by major cloud suppliers entering and investing heavily in the DaaS segment such as Amazon with Amazon Workspaces, Citrix with XenDesktop service and VMware with VMware Horizon DaaS. These DaaS solutions help to bring data and applications go beyond the boundaries of desktops by becoming available on multiple

devices.[7, 20] The following reviews provide the overview of each DaaS solution.

Amazon WorkSpaces provides a fully managed, pay-as-you-go desktop computing solution from the cloud to the end users. Amazon WorkSpaces is a part of AWS ecosystem. Hence, it can easily connect with other AWS products to consolidate the CSU's IT infrastructure. [33]

XenDesktop Service is a service of the Citrix Cloud. The service enables secure access to virtual Windows and Linux desktops. Since Citrix does not have any public cloud or data center as AWS, XenDesktop Service has to operate in its partners' or organizations' IT infrastructure virtualization to deliver DaaS to end users. The service can operate on the public cloud, private cloud or hybrid cloud models. [34]

VMware Horizon DaaS provides a virtual cloud desktop platform for the CSP. The Horizon DaaS platform supports the CSP to deliver virtual workspaces such as full desktops and applications to the CSU on a monthly subscription basis. By partnering with Google and NVIDIA, VMware Horizon DaaS can be tailored to suit the needs to deliver from basic to powerful desktop applications such as CAD and BIM software to any devices, e.g., Chromebook and Android tablets [6].

In this thesis, we concentrate on the Amazon WorkSpaces solution for the implementation. The motivation for choosing Amazon WorkSpaces stemmed from the case company existing infrastructure in AWS ecosystem and the demand to deliver the cloud virtual desktops and the case company application altogether as one package offering. Additionally, with the Amazon WorkSpaces, we are free from handling complex underlying IT infrastructure virtualization as it is manage by the Amazon.

2.4 GPU-Accelerated virtual machines in cloud computing

A graphics processing unit (GPU) is a specialized computer hardware designed to rapidly accelerate the creation of digital images and manipulate computer graphics to display as the visual output. The GPU is an essential hardware component in workstations, especially for graphics-intensive usage.

The case company's software requires high performance graphics capability to function speedily and reliably. Nevertheless, unlike physical desktops, virtual machines cannot utilize the power of a traditional GPU without an emulated graphics adapter. The emulated graphics adapter is usually a software interface that allocates inefficiently the host's graphics processing power

to virtual machines [36]. GPU pass-through, enabled by Intel VT-d/AMD IOMMU techniques [2], can overcome the inadequacy of the emulated graphics adapter by introducing a nearly native graphics performance to virtual machines through exclusive direct I/O accesses without privileged domain involvements. Thus, the GPU pass-through is a favorable virtualization solution to serve graphics-intensive virtual machines in the cloud computing.[22]

NVIDIA GRID is a graphics technology that supports the GPU pass-through for sharing virtual GPUs' resources to multiple virtual machines simultaneously. NVIDIA GRID platform consists of GPU virtualization, remote processing and session management libraries that are capable of supporting concurrent high performance graphics tasks via the network. Besides that, the Computer Unified Device Architecture encapsulated in NVIDIA GRID graphics cards is capable of parallel processing general purpose algorithms in the GPU level that functions as a co-processor to the CPU. Furthermore, NVDIA GRID techniques allow transmitting virtual machines' contents via the cloud service in the form of streams like other media. All these features NVIDIA GRID technology provided makes it become an optimal choice to facilitate the implementation of high performance virtual machines in the cloud. [15, 28]

Amazon Web Services is a prevalent cloud service provider that has taken NVIDIA GRID techniques into production. Amazon Elastic Compute Cloud G2 and Amazon Graphics WorkSpaces are the two renowned virtual cloud computing services that feature high-performance NVIDIA GRID GPUs.[3, 4] Currently, Amazon Graphics WorkSpaces is the solely Desktop as a Service solution that can match the performance requirements from the case computer's software. Hence, this thesis implementation focally engages the Amazon Graphics WorkSpaces DaaS solution.

2.5 Summary

In this chapter, we presented the main concepts and technologies of desktop virtualization. Virtual Desktop Infrastructure, the fundamental idea of desktop virtualization, is scrutinized. In the section 2.2, we investigated thoroughly the cloud computing ontology, its deployment models as well as the cloud computing architecture and services.

After discussing the major cloud computing *as-a-Service* models, we conscientiously explored Desktop as a Service (DaaS), the focal cloud service model that we employ throughout the thesis work. We formulated the consensus on DaaS definition based on different sources [19, 23, 29, 33]. In order to get a more profound understanding of the service, we studied the

underlying service architecture of DaaS. The notable advantages of DaaS for different user groups utilizing the service were discussed. Besides that, we pointed out few limitations that DaaS encounters.

The leading DaaS providers, e.g., Amazon, Citrix and VMware with their explicit DaaS technologies are succinctly described in the section 2.3.4. Furthermore, the high performance GPU utilized in virtual machines in the cloud computing is investigated. The GPU virtualization technique has evolved significantly in the past few years from an inefficient emulated graphics to a nearly native, powerful desktop graphics such as NVIDIA GRID. It is certain that the GPU-Accelerated technique for virtual machines would bring a solid boost for DaaS to reach more users' use cases such as CAD designers or gamers who work with 3D building information modeling software or virtual reality games that often require graphics-intensive machines for heavy 3D rendering tasks. In the next chapter, we delve into the case company's background and their software that we will use to deploy in the Amazon WorkSpaces.

Chapter 3

Case Study Software

This chapter gives a brief introduction about the case company. One of the case company's core software products, Tekla Structures, is described. Then, we delve into the case company's continuous delivery system. Furthermore, the prerequisite software and tools revolving Tekla Structures are concisely presented.

3.1 Case Study Company

Trimble Inc. is a global corporation that provides integrated information technology solutions for a wide range of industries such as agriculture, construction, geospatial, transportation and logistics. Trimble's solutions are being used in more than 150 countries with offices in 35 countries. [37]

Tekla Corporation was acquired and became a part of Trimble Inc. in 2011. The addition of Tekla's BIM software has further enhanced Trimble's building and construction industry. Tekla Structures is the main software offering in the case company's structures portfolio. [37]

3.2 Tekla Structures Software

Tekla Structures is an extensive 3D Building Information Modeling (BIM) software. Tekla Structures provides a wide range of configurations and localized environments to fulfill various customer requirements. The software allows structural engineers, fabricators and construction professionals to create, manage and share constructible information throughout the design-build-operate life cycle [37]. Tekla Structures can be utilized to design complex 3D models. The software is fully capable of handling large size models such

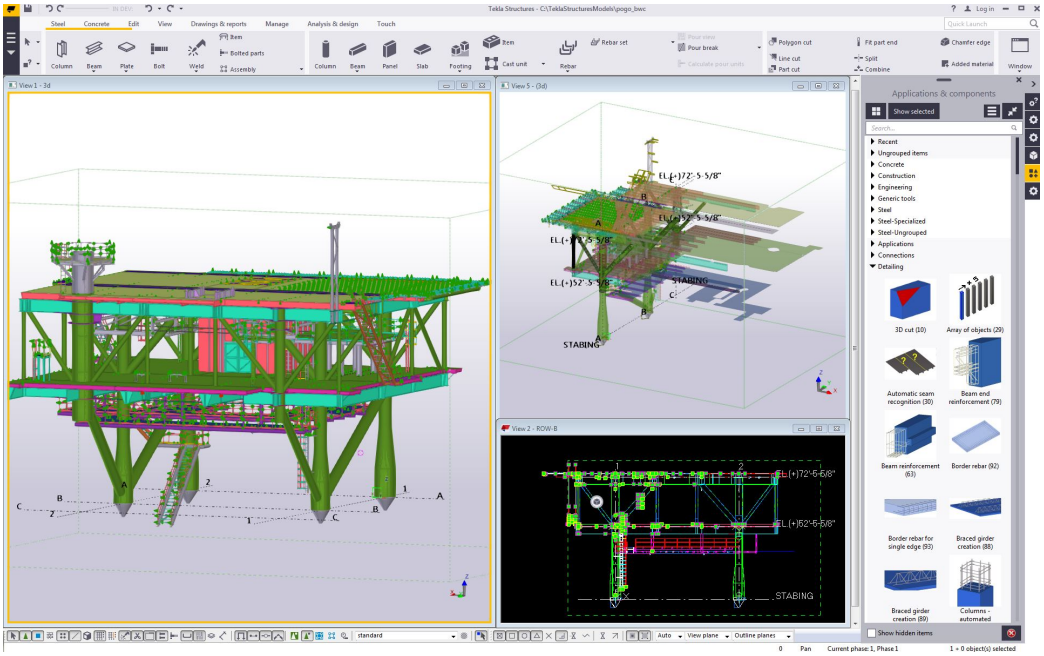


Figure 3.1: Tekla Structures Screenshot

as skyscrapers and stadiums that comprise millions of objects and components. Figure 3.1 illustrates a steel and concrete offshore platform modeled with Tekla Structures. Furthermore, the software supports structural engineers to manipulate 3D models to generate 2D drawings and reports for references and manufacturing. The general arrangement drawings from Tekla Structures are regularly produced to provide information of model views and erection elevation in the construction site.

Tekla Structures is a mature software. The software development has been progressing for more than 26 years with the code base consisting of millions lines of code divided into over a hundred logical libraries. Tekla Structures endorses object oriented design and was initially developed in C programming language. Later on, C++, C# and the .NET framework were utilized for easier development and better interoperability with other applications. The case company's developers are constantly working on the software to develop new features, fix defects, refine the software architecture and enhance the performance.

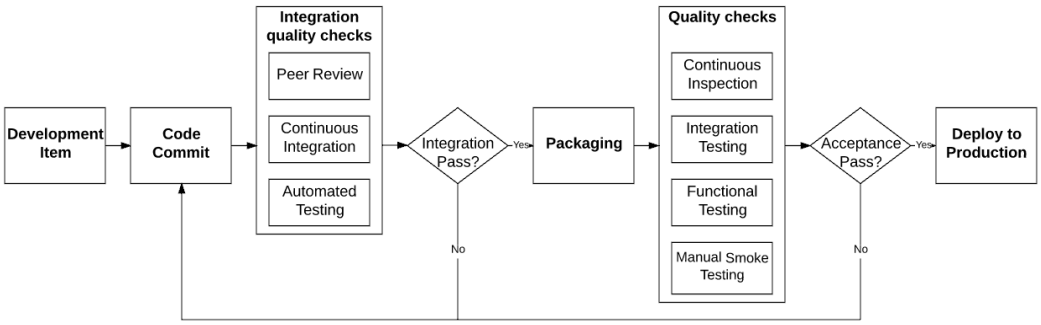


Figure 3.2: Tekla Structures continuous delivery system

3.3 Continuous delivery system

Continuous delivery (CD) is a software engineering approach that automates building and testing a software product on a regular basis and maintains the releasable state of the software at any point in time. In other words, CD approach presents a fully automated pipeline from code commit to production artifacts.[8, 16] The case company has adopted CD practice for the past few years. Figure 3.2 depicts the case company's CD pipeline for Tekla Structures software.

In the initial step, development items logged in the case company's issue management system are developed. Once developers commit the code to the software repository in the case company version control system (VCS), this code check-in automatically triggers the continuous integration (CI) system that compiles the source code and executes unit tests. If the code commit stage fails the integration checks, the CD pipeline halts and alerts the developers. Developers can then investigate the error logs from the CI system, revise the code, get peer review approved and push the code to the mainline repository. It again triggers the integration quality checks. If all the tests are passed, the pipeline flows to the next stage.

The packaging stage automatically builds artifacts for the acceptance testing. Specifically, these artifacts include the software installer and its binaries generated from the previous stage. After successfully created, these artifacts are uploaded to the central repository for further tests and distribution.

Quality checks is the next step in the CD pipeline. This stage thoroughly analyzes the software so that it meets the company standards for the production. Continuous inspection holistically measures the code quality based on rule-based defect identification. If the code quality is below an acceptable threshold, developers are reported about the quality flaws. Developers can then correct the defects while the impacts are manageable and the fix is still

viable. Integration test examines problems that arise when combining units into modules and modules together. This test helps to verify that different parts and components of Tekla Structures software can work properly together as a unified entity. The case company has an in-house developed system that is utilized for functional and regression testing. For each TS version, exhaustive amount of test cases are carried out systematically in various TS models in testing agents. The functional testing system does not aim to find out all defects but to give assurance that the tested TS version reaches the acceptance levels for deployment. Even though the automated testing is genuinely comprehensive, manual smoke testing remains essential as a part of the quality checks. In the manual smoke tests, manual testers execute exploratory inspections with different user roles and interact with the software as customers would do on their daily work. Usability and user experience tests are also performed by business owners and selected customers to perceive how they appraise the software. When all these tests are completed satisfactorily, the artifacts are switched from testing mode to release mode. At this stage, the software has passed the quality checks and can be deployed to the production.

Applying CD approach has brought several benefits for the case company. Productivity and efficiency in the release process have enhanced substantially. In the old release process, the source code was compiled and the software was packaged once in each development cycle. Now, the software can be released several times daily with minimal human intervention. Besides that, development-operation engineers do not need to spend ample effort on troubleshooting errors caused by the former practice as the CD pipeline can eliminate these manual arduous configurations. Moreover, the CD system helps developers to be more agile and flexible in delivering new features and fixing defects. For example, whenever a new feature is developed, it can be immediately pushed to the main repository that triggers the CD pipeline for compiling, integration and functional tests. As the result, developers receive the report and feedback about their new development faster so as to have a better plan to get the product adjusted and released to customers.

3.4 Pre-requisite Software

3.4.1 Tekla Structures License Administration Tool

Tekla Structures requires a valid license to operate. The licenses are managed by FlexNet [1] Publisher License Management licensing system and stored in a

[1]FlexNet is a Flexera Software's licensing system

trusted storage. Tekla Structures License Administration Tool (LAT) is built on top of the activation-based FlexNet licensing system. LAT is utilized to save the entitlement certificate and activate the licenses that can be applied to Tekla Structures software in client computers.

The major advantage of LAT is that TS licenses are not permanently linked to a MAC address or password files. Hence, customers can easily update and renew the licenses. Moreover, these licenses can be flexibly administered. LAT allows customers to activate licenses based on their needs, for instance, some licenses can be activated to a common license server in a local area network while others on specific users' workstations.[38]

3.4.2 Tekla Structures Environments

Due to the nature of the construction industry that requires country and region-specific standards and profiles for modeling and drawing structures, Tekla Structures (TS) provides different environment packages based on definitive structural design standards to fulfill the construction's code of conduct. At the moment of this writing, 32 TS environments are being provided. Each TS environment consists of tailored settings, material grades, variables, reports and templates. Customers can have many environments installed in TS, at least one environment installed is required for TS operation.

Chapter 4

Amazon Web Services and other tools

Amazon Web Services (AWS) provide IT infrastructure services to enterprises globally in the form of a web-based cloud computing. The main benefits derived from AWS for the case company are the multiple regions availability and the comprehensive, flexible IT cloud infrastructure Amazon provided.

In the following sections, we discuss the Amazon Web Services offered by Amazon that are utilized in the thesis implementation. The sections' order is based on the significance of these services utilized in the implementation. Furthermore, we examine other Windows application software used as a part of the implementation.

4.1 Amazon WorkSpaces

Amazon WorkSpaces is a fully managed desktop computing service that operates in the AWS cloud. End users are able to easily manipulate WorkSpaces, i.e., cloud-based desktops in Amazon WorkSpaces from various devices such as computers, tablets and smartphones with the Internet access. Additionally, the existing IT systems and enterprises' on-premises Active Directory can be fully integrated with Amazon WorkSpaces in order to provide a consistent access to the enterprises' resources. Cloud service users can pay hourly rate for each hour they use WorkSpaces or a fixed monthly fee based on the WorkSpaces used and their hardware configurations. [26]

This thesis implementation revolves around the Amazon WorkSpaces service. Firstly, we delve into Amazon WorkSpaces Application Manager (WAM) for packaging the virtualized case company software, Tekla Structures, in order to be operated on WorkSpace cloud desktops. Secondly,

WorkSpaces delivery process is studied to find out the most efficient and feasible approach to deliver continuously a bundle of the high-quality WorkSpace cloud desktops and Tekla Structures to end users.

4.1.1 WorkSpaces

WorkSpaces are fully managed virtual desktops spawned from Amazon WorkSpaces that run on the AWS cloud. The WorkSpace is the virtual desktop that end users will connect to and work with. Based on the requirement, the end users can choose between four different hardware configurations: Value, Standard, Performance and Graphics. In this thesis work, the Graphics WorkSpace is exclusively utilized as it offers a high-end GPU-powered virtual desktop that fulfills the Tekla Structures' hardware requirements. Furthermore, the NVIDIA K520 graphics card used in the Graphics WorkSpace supports OpenGL 4.x, DirectX, CUDA and OpenCL that are essential for rendering processes in Tekla Structures. Below is the specification of the GPU-powered Graphics WorkSpace [4, 26]:

- **Processing** - Intel® Xeon® E5-2670 @ 2.60 GHz, 8 virtual CPUs.
- **Memory** - 15 GB.
- **Graphics** - NVIDIA GRID K520, 4 GB memory.
- **Display** - Support max. 2 monitors with 2560x1600 pixels resolution.
- **System volume** - 100 SSD GB.
- **User volume** - 100 SSD GB.

4.1.2 PC-over-IP Protocol for Virtual Desktop Delivery

Amazon WorkSpaces adopts Teradici's PC-over-IP (PCoIP) protocol technology to deliver virtual desktops to client devices. The PCoIP remote display protocol is a nimble and secure communication channel since it only transfers images built up from pixels' location information. In essence, only encrypted pixels of images are transferred between client devices and their WorkSpaces while the actual data always remains safely in the AWS data center. In addition, the PCoIP protocol supports high resolution and multiple displays that would fulfil the case company's software usability. [24, 26]

4.2 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (EC2) is a virtual computing environment that provides scalable computing capacity over the cloud. Amazon EC2 can launch various EC2 instances with different configurations and settings such as operating systems, custom applications and network access permissions. Since new Amazon EC2 instances can be acquired and run within a short time, it facilitates the cloud service users (CSU) to flexibly scale the computing capacity as the requirements alter. [26]

In the implementation, we use two specific configurations of Amazon Machine Image named: Amazon WAM Admin Studio AMI (WAM Studio) and Amazon WAM Admin Player AMI (WAM Player). The WAM Studio is utilized to package the software as a virtual container so that it can be used in Amazon WorkSpaces. After the software is packaged, the WAM Player does the validation for this package to verify its readiness in the WorkSpaces.

4.3 Amazon Simple Storage Service

Amazon Simple Storage Service (S3) is a web-based cloud service that supplies secure and scalable data storage. Amazon S3 enables CSU to store and retrieve unlimited data from the cloud. There are different options for choosing the most appropriate storage classes based on the data access frequency. [26]

We employ Amazon S3 to transfer the Tekla Structures software and other data from the case company on-premises database and AWS ecosystem itself for the automation prototype.

4.4 AWS Lambda

AWS Lambda enables code to be executed in response to events such as Amazon resources change or custom events triggered by other services. Lambda can run code for most application types and back-end services without administration and managing servers. [26]

Lambda functions are developed in the prototype to create, monitor and terminate the desired EC2 instances as a part of the packaging automation for the Tekla Structures software as well as calling the Amazon WorkSpaces API for launching new WorkSpaces. All the Lambda functions are written in Python 2.7.

4.5 Amazon API Gateway

Amazon API Gateway is a management service to create, publish and manage Application programming interfaces (API). An API in Amazon API Gateway allows other applications to access data or functionality from the back-end services such as AWS lambda functions. Amazon API Gateway can supervise all API tasks including access and authorization control, traffic handling and API version control. [26]

In the implementation, we use Amazon API Gateway in conjunction with Amazon Lambda functions to create an AWS serverless service.

4.6 AWS Directory Service

AWS Directory Service gives administrators the capability to connect AWS resources with an on-premises Microsoft Active Directory (AD) or to create a standalone AD in the AWS cloud. Connecting to an existing on-premises directory can be implemented through AD Connector gateway. On the other hand, a brand new Microsoft AD supporting up to 50000 users can be set up with AWS Directory Service.[26] We will use AWS Directory Service to create a new simple AD to manage Amazon WorkSpaces users.

4.7 AWS Identity and Access Management

AWS Identity and Access Management (IAM) provides a web-based management system to administer AWS services and resources for AWS users and groups. IAM is used throughout the whole implementation to assign suitable access permissions to different AWS resources.

4.8 Other applications

Besides the services utilized in the AWS ecosystem, the implementation requires some additional scripts and tools to fully automate the packaging process in Amazon WorkSpaces Application Manager (WAM). The following subsections introduce these applications.

4.8.1 SikuliX

SikuliX is a Java application that uses image recognition to automate anything displayed on the desktop screen. SikuliX can control peripherals to

interact with the identified GUI Windows items. Java implementation of Python also known as Jython is used to develop SikuliX scripts. [14]

The motivation to use SikuliX stemmed from the lack of an available Amazon WAM API to handle the packaging tasks nor access to the Amazon WAM internal source code. Hence, SikuliX would be the appropriate automation tool in this scenario.

4.8.2 InstallShield

InstallShield is a Windows packaging software. In the implementation, InstallShield is used to create an auto-install installer package that contains Java run-time binaries and SikuliX scripts. The installer will be installed automatically right after the EC2 instances are successfully created and initiate the SikuliX scripts.

4.9 Summary

This chapter presented the Amazon Web Services and tools we use to develop the prototype in the implementation part. Particularly, we studied the Amazon WorkSpaces in detail and discussed about the WorkSpace, a virtual machine unit in the Amazon WorkSpaces. The WorkSpace with the Graphics hardware configuration is also reviewed. Additionally, we investigated the PC-over-IP protocol and its advantages for being used in the Amazon WorkSpaces.

Amazon API Gateway and AWS Lambda utilized to form up an AWS serverless service in the prototype were examined. Other AWS services necessity to build the prototype were presented and their roles in the system were described. All these services are essential to build the working prototype as each of them plays a specific role in the system. Other tools such as SikuliX and InstallShield were used to create the graphical scripts and package the scripts and Tekla Structures installer as one package for the automation.

The detailed usage and functionality of these AWS services and tools are scrutinized in the next chapters.

Chapter 5

Design

This chapter presents the architectural design that is used to develop the prototype in the implementation. We explain the overall processes and the work-flow of the prototype design.

5.1 Functional objectives

The main objective of the implementation is to automate the whole deployment process of the case company software in Amazon WorkSpaces to the greatest extent. In order to achieve the objective, we study the requirements to accomplish the deployment process. We then investigate which tasks in the deployment process can and should be automated. Thenceforth, the prototype design to automate these tasks is proposed and discussed.

5.2 Amazon WorkSpaces Deployment Process

Figure 5.1 illustrates the complete high-level deployment process of a Windows software, e.g., Tekla Structures in the Amazon WorkSpaces. Firstly, the software needs to be packaged exclusively for Amazon WorkSpaces[1]. The packaging operation consists of virtualizing and validating the software. The WAM Studio EC2 instance that includes Amazon WAM Admin Studio tool is utilized to virtualize the software. After the virtualized application container of the software is created, the WAM Player EC2 instance is launched to

[1]Amazon WorkSpaces provisions a Windows application as a virtualized application container.

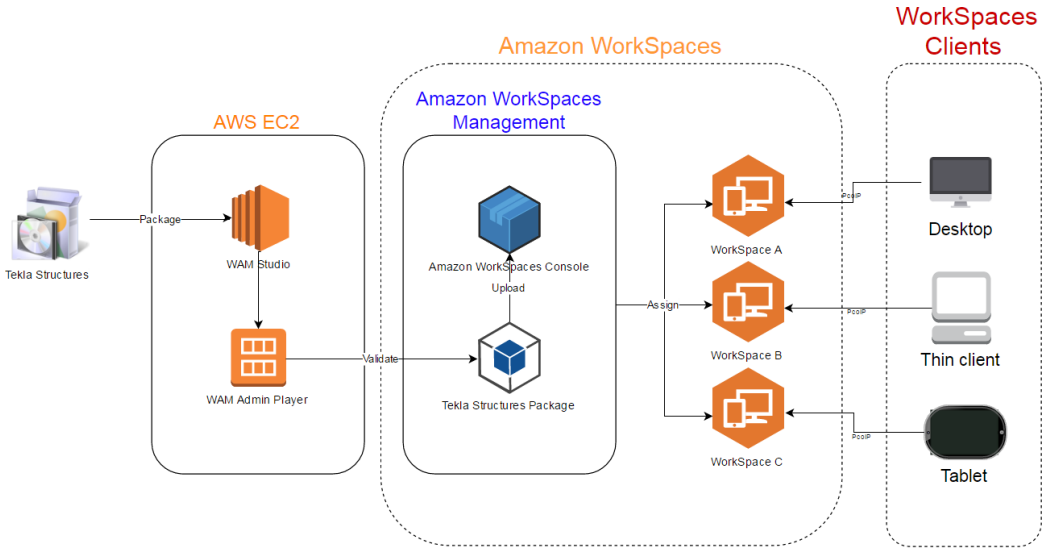


Figure 5.1: Tekla Structures deployment process in Amazon WorkSpaces

validate the package functionality in the Amazon WorkSpaces environment. If the package is validated without errors, it will be released and visible in Amazon WorkSpaces Console. At this stage, the Tekla Structures virtualized package is ready to be assigned to the WorkSpaces virtual desktops. The package would behave as a standard Windows application even though they are actually not installed in these WorkSpaces. End users can access to their dedicated bundle of WorkSpaces with the Tekla Structures installed via Amazon WorkSpaces client. The Amazon WorkSpaces client is responsible to transmit the user's input such as keyboard, mouse and touch to the WorkSpace in the data center. Furthermore, the encrypted image pixels and audio sent from the WorkSpace to the user are decrypted, decompressed and displayed seamlessly via the PCoIP protocol.

As described in the previous paragraph, there are 4 major procedures in the deployment process: virtualizing, validating, uploading and launching. The uploading procedure proceeds automatically by Amazon WAM. Hence, there are 3 remaining procedures we need to automate in order to achieve a continuous deployment system in the Amazon WorkSpaces service. Virtualizing and validating are grouped together as a packaging operation. One automation system is developed for the virtualizing and validating procedures due to their similarity in resources and correlative in actions. The launching procedure is the second operation we aim to automate. The following sub sections provides the in-depth system design details for these 2 operations.

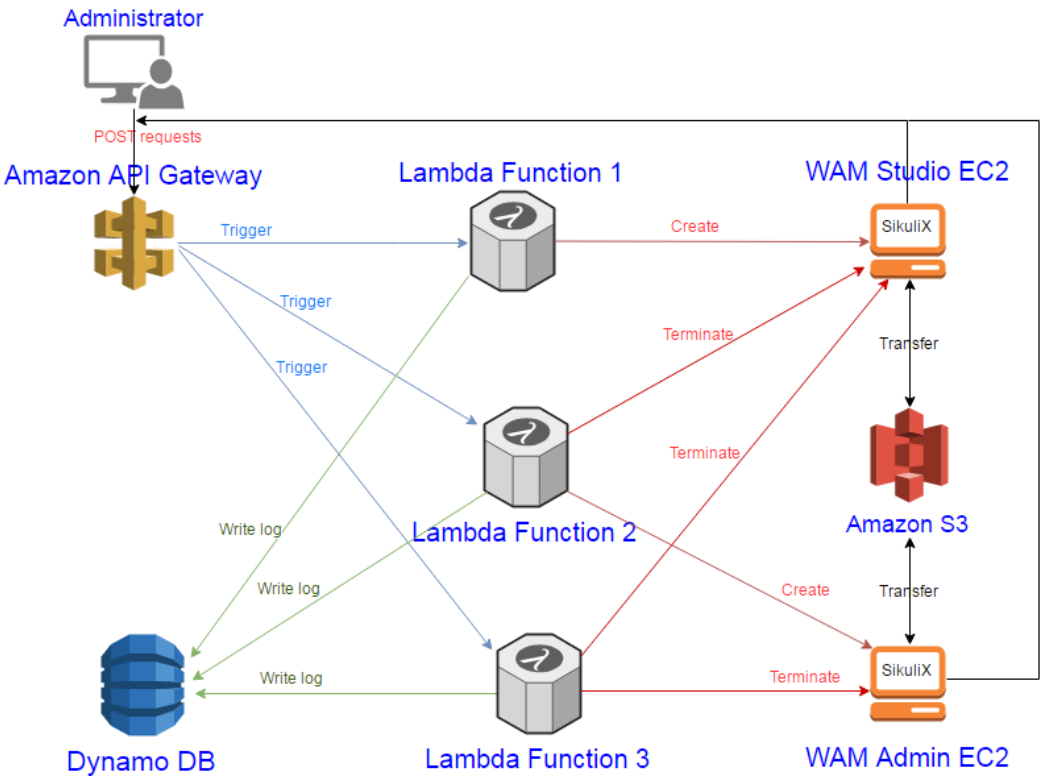


Figure 5.2: Tekla Structures packaging workflow in Amazon WorkSpaces

5.2.1 Packaging application in Amazon WorkSpaces

Figure 5.2 demonstrates the packaging operation's system architecture in the deployment process. The system involves principally five AWS services: Amazon API Gateway, AWS Lambda, Amazon EC2, Amazon Dynamo DB and Amazon S3. By combining AWS Lambda with Amazon API Gateway, we aim to build a serverless system without administrative provisions.

The figure 5.2 also illustrates the system overview. Initially, administrators make a Representational state transfer (REST) call to an endpoint created by Amazon API Gateway. Based on the call request's content, Amazon API Gateway would trigger the corresponding function built in AWS Lambda. For instance, the Lambda Function 1 would launch a WAM Studio EC2 instance with a pre-defined configuration. Once the instance is successfully launched, the Lambda Function 1 commands the instance to download the SikuliX script and Tekla Structures installation files from Amazon S3. When the files download completed, the SikuliX script automatically starts to virtualize the software in the WAM Studio. After the software has been virtualized, the instance sends a POST request to Amazon API Gateway that

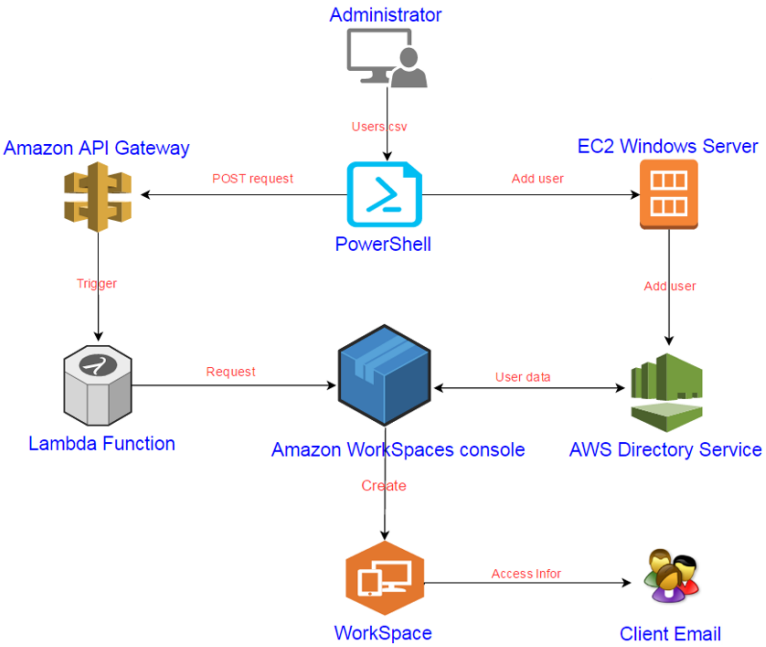


Figure 5.3: Workflow for launching Amazon WorkSpaces to end users

provokes the Lambda Function 2. The Lambda Function 2 would terminate the WAM Studio EC2 instance as it is no longer needed. Concurrently, the Function 2 creates a new WAM Player EC2 instance. The second SikuliX script is downloaded and executes automatically to validate the virtualized software package. Once the package is tested successfully, it will be available in the WAM Console, ready to be utilized. Another POST request is sent from the WAM Player to notify the validation is completed. This POST request triggers the Lambda Function 3 that would terminate all running EC2 instances and verify the operation accomplishment. Additionally, all lambda functions write log files of their operational status into a table in Amazon Dynamo DB for monitoring and analyzing. Administrators can also interfere and manipulate any lambda functions from a special GUI tool developed in this implementation.

5.2.2 Launching Amazon WorkSpaces to end users

The second operation is the new WorkSpaces launching procedure. It consists of adding new users to a Microsoft Active Directory (AD), creating WorkSpaces and providing WorkSpaces' accessed information to end users. Figure 5.3 illustrates the second operation's system architecture in the deployment process.

As can be seen in Figure 5.3, we use the same serverless service framework with the Amazon API Gateway and the AWS Lambda as in the first operation. Firstly, administrator sends a user accounts' database consisting of usernames and their personal information, e.g., full names, display name and emails to a PowerShell script. The script will import the database and send commands to add each username and its information to the AD. Thereupon, the script sends POST requests that contains the authentication token and the usernames to the Amazon API Gateway. The API Gateway will trigger the Lambda Function and deliver the POST content to the Lambda Function event handler. The event handler will verify the authentication token. If the token is correct, the Lambda Function sends WorkSpace launching requests with the usernames to Amazon WorkSpaces Console. Amazon WorkSpaces Console (AWC) will start the WorkSpace creation process with the configuration parameters received from the Lambda Function request. Initially, AWC has to ratify the received usernames with the matching ones in the AD through the AWS Directory Service. If the username is valid, AWC will enquire the user's email, last name and first name and display name from the AD and launch a new WorkSpace based on these values. When the WorkSpace is ready, end users will be notified with the username and an accessible link to connect to their WorkSpace via email. End users can access WorkSpaces anytime by Amazon WorkSpaces client.

Chapter 6

Implementation and System Prototype

We develop the system based on the designs proposed in the Chapter 5. The implementation follows these two major operations according to the designs: packaging in Amazon WorkSpaces and launching WorkSpaces to end users.

We apply agile software development approach for this implementation [25]. Our ultimate goal is to quickly build the working system and start to study the system feasibility to apply for the case company production purpose. In essence, we aim to create a fundamental system without advanced features yet. If it proves to be beneficial, we can keep the continuous improvement in the future development phases.

In order to reduce data latency and match the system similarly to the real-life production scenario, all Amazon Web Services that we deploy are in the EU (Ireland) data center in the *eu-west-1* region. This AWS data center is also in the closest proximity to the case company. It should also be taken into account that most Amazon data centers are not interconnected between different regions. Hence, the AWS system employed in one region is not available in other regions [26].

6.1 Packaging application in Amazon WorkSpaces

The implementation of the first operation is divided into small components. Each component is responsible for a specific task. When all the components are developed, we integrate all of them into one system for the operation. The following subsections describe the required components and their roles

in the system.

6.1.1 AWS IAM component

In order to access AWS services, we need to configure the permissions in AWS IAM. The AWS IAM can grant policies on specific actions and resources of any AWS services that users can administer. Below is an example policy for full control permission in Amazon WorkSpaces.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "workspaces:CreateWorkspaces",
                "workspaces:DescribeWorkspaces",
                "workspaces:RebootWorkspaces",
                "workspaces:RebuildWorkspaces",
                "workspaces:TerminateWorkspaces",
                "workspaces:DescribeWorkspaceDirectories",
                "workspaces:DescribeWorkspaceBundles",
                "workspaces:ModifyWorkspaceProperties",
                "workspaces:StopWorkspaces",
                "workspaces:StartWorkspaces",
                "workspaces:DescribeWorkspacesConnectionStatus",
                "workspaces:CreateTags",
                "workspaces:DeleteTags",
                "workspaces:DescribeTags",
                "kms:ListKeys",
                "kms:ListAliases",
                "kms:DescribeKey"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

For the experimental purpose of this implementation, we give ourselves Administrator Access permission that can exploit all the actions and resources in AWS services.

6.1.2 Amazon API Gateway component

In Amazon API Gateway, we create a new API named WorkSpacesAPI. In this new blank API, we generate a resource that would expose the REST service. Then, we create 3 POST methods on the WorkSpacesAPI resource

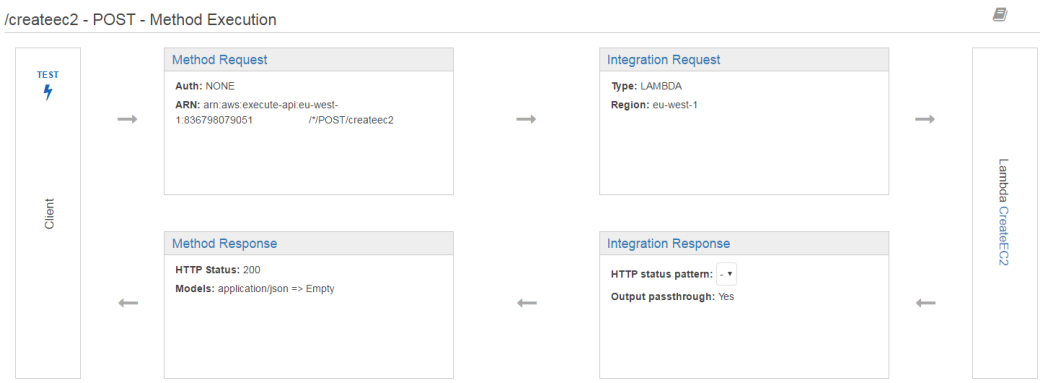


Figure 6.1: Amazon API Gateway Console for WorkSpacesAPI

and connect them to these 3 corresponding Lambda functions. If the POST request is called correctly, it will invoke the inquired Lambda function and return to the caller the response status. Figure 6.1 depicts the workflow of the first POST method that trigger Lambda Function 1. At last, we deploy the API to be available in the public network so that it can be called outside of the AWS by any REST clients.

The POST methods are tested with a Python API client code below. The request body contains a 'token' JSON object. The token value is compared with the correct one in the Lambda function. If the token value is matched, the Lambda main function will proceed.

```
1 def request(InvokeURL):
2     r = requests.post(InvokeURL,data =json.dumps({'token':'key'
      }))
3     return r.text
```

6.1.3 AWS Lambda component

We develop 3 separate AWS lambda functions for the first operation. The lambda functions are written in Python 2.7 and work as described in the section 5.2.1. We use boto3 module[1] as the AWS software development kit (SDK) in Python 2.7. boto3 is an intensive module that provides object-oriented and low-level direct service access to several AWS. Since AWS Lambda can run code without provisioning and administration, we have to create a specific AWS IAM role for these 3 lambda functions to limit their access to explicit services, namely, Amazon EC2, Amazon Dynamo DB and

[1]http://boto3.readthedocs.io/en/latest/reference/services/index.html

Amazon API Gateway. After setting up the AWS lambda configuration, the AWS lambda functions can respond to events from the Amazon API Gateway and automatically trigger other AWS services accordingly.

6.1.4 Amazon S3 component

Amazon S3 is utilized to store SikuliX scripts and Tekla Structures installers that are used by WAM EC2 instances. We create a bucket, i.e., folder in Amazon S3 and grant the read permission for the Lambda functions. Server-side encryption with Amazon S3-Managed Keys is adopted to encrypt the data so that only authenticated entities having access permissions can modify the data [30].

6.1.5 Amazon EC2 component

As presented in the Design chapter, there are two dedicated Amazon EC2 instances utilized in the first operation: WAM Studio and WAM Player. A predefined IAM role *AmazonWamAppPackaging* is assigned to these instances so that they can access Amazon WAM administrative features and the WAM application catalog. These instances are launched with all the configurations designated in the lambda functions. Upon the instance creation, the lambda functions will trigger a PowerShell script to download the required files from Amazon S3 and run the corresponding SikuliX scripts.

6.1.6 SikuliX scripts component

Two SikuliX scripts are written to graphically automate the installation process of virtualizing the software for Amazon WorkSpaces in the WAM Studio and the validation executed in the WAM Player. The SikuliX scripting language is built on Jython[2] and requires Java Runtime environment to function. The SikuliX unique feature lies on its capability of assigning captured screen images as values to variables.

To create SikuliX scripts for these two processes, we first implement the packaging and validation tasks manually in the WAM Studio and the WAM Player and capture all the taken actions. Then, we write SikuliX scripts that guide the executions step-by-step based on the captured images. Furthermore, functions to handle exceptional conditions and errors are defined. Figure 6.2 shows a code snippet of our SikuliX script in the SikuliX IDE.

[2]http://www.jython.org/

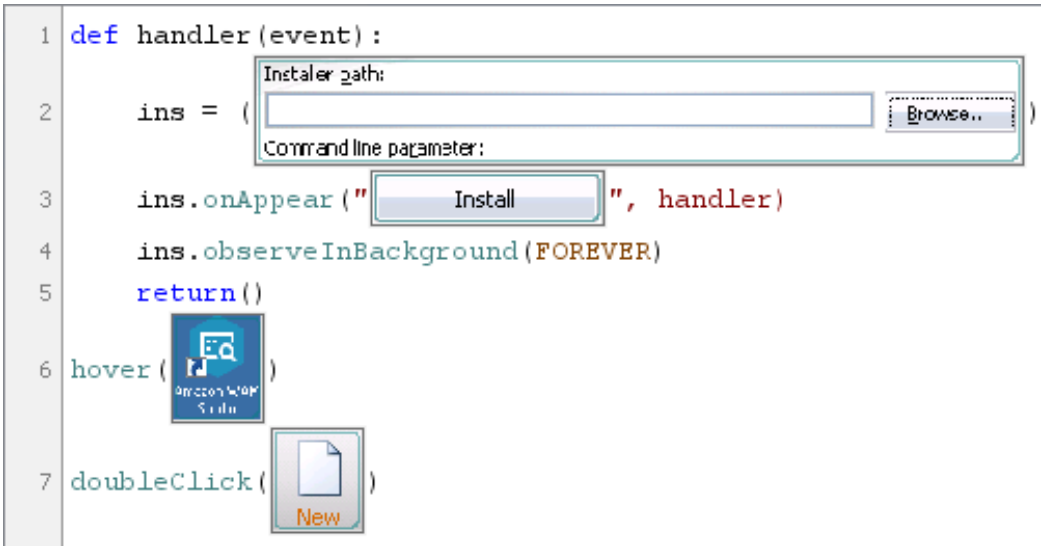


Figure 6.2: SikuliX script snippet for automating tasks in Amazon WAM Studio instance

6.1.7 Amazon DynamoDB component

We build a DynamoDB table that records the operation status. Initially, we create a blank table named *WorkSpacesAPI*. The table contains an attribute item *Tag* that is a primary key and has the value *WorkSpaces*. Other items includes:

- ID: indicates a Lambda function number
- Description: shows a Lambda function information
- Timelog: records time stamp when a Lambda function is triggered
- UUID: universal unique ID for a triggered Lambda function

The Lambda functions write data logs to the WorkSpacesAPI table whenever they are in operation. Administrator can query the table for information from Amazon DynamoDB dashboard as shown in Figure 6.3 or with the AWS SDK.

6.1.8 Associate all components as one system

When all the components function as expected, we consolidate these closely connected components into one system that can automate the first operation. Moreover, an administrator tool for the virtualizing and validating operation

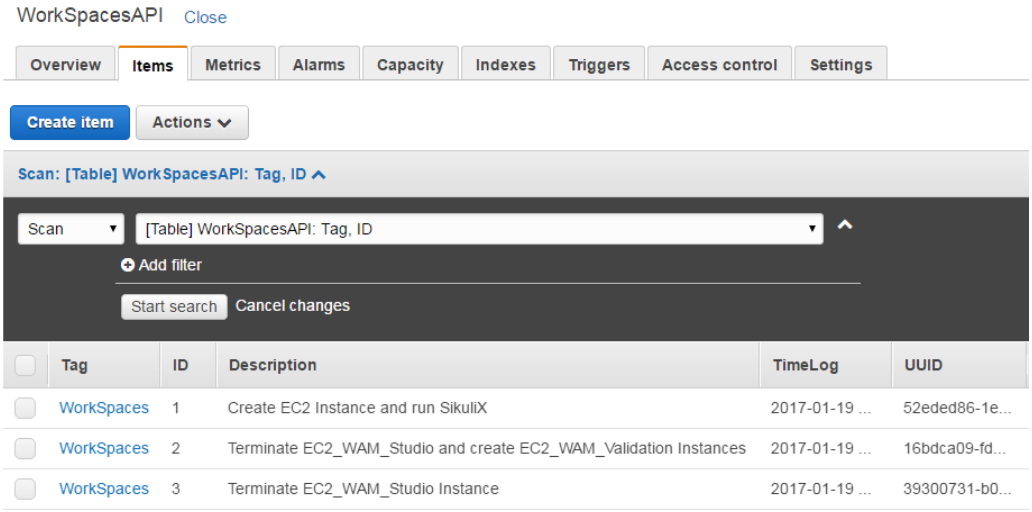


Figure 6.3: WorkSpacesAPI table query result in Amazon DynamoDB dashboard

is created. The tool allows administrators to check for the operation status and trigger different Lambda functions at any given time.

Besides that, it is possible to integrate the first operation system to the case company's continuous delivery system in the TeamCity. One more build step can be added to TeamCity pipeline so that it will automatically upload the released software artifacts, e.g., Tekla Structures installers to the Amazon S3. TeamCity can then replace the administrator to send a POST request that triggers the first operation to start the software packaging automation.

6.2 Launching WorkSpaces to end users

As in the first operation, we divide the second operation into small components. The following subsections describe the required components and their roles in the system.

6.2.1 Amazon API Gateway component

We make use of the existing WorkSpacesAPI from the first operation in the Amazon API Gateway. A new resource that would expose a new Lambda function via the POST method is created. After being tested internally, we deploy a new stage to the WorkSpacesAPI so it is accessible over the Internet.

6.2.2 AWS Lambda component

We develop the new AWS Lambda function for the second operation. The lambda function will retrieve *token* and *username* value from the POST request via Amazon API Gateway. The *token* will be verified. If it is matched, the Lambda main function will be initiated to send a low-level client request with the given *username* to Amazon WorkSpaces Console that will launch a new WorkSpace accordingly. This Lambda operation is asynchronous and returns before the WorkSpace is created.

6.2.3 AWS IAM component

Besides the permissions inherited from the first operation, we grant the new Lambda function an additional admin permission to control the Amazon WorkSpaces resources. Furthermore, a Windows Server EC2 instance is given full access to Simple System Manager (SSM) in order to administer AWS Directory Service.

6.2.4 AWS Directory Service component

AWS Directory Service is utilized to create an Active Directory (AD). The Simple AD fulfils the experimental purpose of this thesis work as it equips with a subset of Microsoft AD features such as user accounts and group memberships management, group policies control and capability for Kerberos-based single sign-on [31]. We set up a Simple AD powered by Samba 4 Active Directory Compatible Server[3]. This Simple AD is utilized to manage the groups and user accounts information for Amazon WorkSpaces in the active directory domain "Trim.ble.com".

6.2.5 Amazon EC2 component

We launch a dedicated Microsoft Windows Server 2008 EC2 instance to manage Active Directory Domain Services (ADDS). SSM Config [32] is utilized to specify the "Trim.ble.com" domain join details and associate the SSM document to the EC2 instance. After the EC2 has successfully joined the domain, ADDS and AD Lightweight Directory Services Tools are installed and configured. In Active Directory Users and Computers, we can manage users and groups in the Simple AD that are used by the Amazon WorkSpaces. Alternatively, users and groups can be administered with PowerShell commands

[3]https://wiki.samba.org/

via Systems Manager Services in the Amazon EC2 Dashboard or AWS command line interface. The SSM Config allows sending these commands from the domain to the local EC2 server instance.

6.2.6 Amazon WorkSpaces console

In this second operation, Amazon WorkSpaces console is primarily utilized to verify user accounts with the AWS Directory Server and launch new WorkSpaces with the users' information gathered from the AD. These tasks are defined in the Lambda function and triggered automatically upon the Lambda request. The GPU-Powered Graphics configuration is applied for new WorkSpaces. The WorkSpace creation process takes approximately 20 minutes to complete. Upon completion, Amazon WorkSpaces console will send the email that contains the WorkSpace log in information to the user.

One WorkSpace is dedicated only to one user all the time. The user can access to their WorkSpace via Amazon WorkSpaces client. As Amazon WorkSpaces uses PCoIP protocol to transfer the pixels from WorkSpaces to the client, TCP/UDP port 4172 has to be opened so that PCoIP connection can be established. We will thoroughly review Amazon WorkSpaces console and its other features in the next chapter.

6.2.7 Associate all components as one system

In the second operation, each component is interconnected to one another. Subsequently, we join all the components to assemble one coherent system for the second operation. We also checked if the integration between different components works well together and there is no conflict emerged.

6.3 Summary

By the end of this implementation, we have successfully created a functional prototype that can automate the software packaging and delivery processes. As our goal for this implementation is to create a fundamental working system, some features can be added and improved in the future development. For instance, we can add more specific function to check for failure, exceptional cases and manage iteration tasks. In the next chapter, we delve into detail assessment about the system prototype developed in this chapter.

Chapter 7

Evaluation

This chapter evaluates the system prototype built in the implementation from administrators' perspectives. Additionally, we discuss in-depth the usage of Amazon WorkSpaces Application Manager tools and console with Tekla Structures. Furthermore, we carry out different Tekla Structures usage scenarios to evaluate the performance of Tekla Structures and its interoperability with other software in Amazon WorkSpaces.

7.1 The prototype experiment and analysis

In order to evaluate the prototype, we appraise the two operations provided from the implementation. In the first operation, we will package the case company, Tekla Structures Learning edition, based on the proposed chronological order. Then, we follow the second operation procedure to create WorkSpaces to two different end users and assign the virtualized Tekla Structures to them.

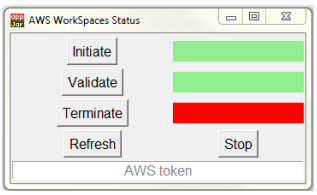


Figure 7.1: AWS WorkSpaces Operation Status GUI

7.1.1 Virtualizing and validating Tekla Structures for Amazon WorkSpaces

Initially, Tekla Structures installer, e.g., TeklaStructuresLearning.exe is uploaded to Amazon S3 bucket via TeamCity, the case company's continuous delivery system. We then use the tool developed for the first operation shown in Figure 7.1 to send a POST request containing a verification token to the Amazon API Gateway. In the underlying scene, the Amazon API Gateway triggers the first Lambda function that verifies and launches a WAM Studio EC2 instance, downloads Tekla Structures installer from the S3 bucket and deploys the SikuliX script. Figure 7.2 captured a screenshot of the WAM Studio being automated by the SikuliX script to virtualize the software. Since the Tekla Structures installer we used is not built natively for the virtualization environment, there are some fine tunings that need to be performed in the Amazon WAM Admin Studio before delivering the package for validation:

- Folders: Tekla Structures uses custom keyboard shortcuts configuration located under folder "*%LOCALAPPDATA%\Trimble\TeklaStructures\2016\Settings*". The folder is not created during the package container creation. However, when Tekla Structures operates, it requires this folder to save the default configuration file. Hence, we have to add *Settings* folder in the WAM Admin Studio Files tab so that this folder will be permanently created in the first place.

- Fonts: A variety of application-type fonts, for example, fontfile.fon is utilized by Tekla Structures. When fonts are being evaluated by the WAM Player, by default they are being made available for system-wide use. However, only true-type fonts (fontfile.ttf) can be installed for the system use not application fonts (fontfile.fon). Consequently, the validation of Tekla Structures is failed since *.fon files cannot be installed to Windows font repository. To solve this issue, we have to exclude the application fonts in the WAM Admin Studio Fonts tab. Fonts will still be available to the application as they are remained in the proper path: *[ProgramData]\Tekla Structures\2016\Environments\common\fonts*

We add these file and folder tunings implementation in the SikuliX script so it will be configured automatically. Once the virtualized package is created, the SikuliX script starts the Windows PowerShell [1] script that will send a POST request to activate the second Lambda function.

The second Lambda terminates the WAM Studio and launches the WAM Player for validation. The second SikuliX script is executed in the WAM

[1] https://msdn.microsoft.com/en-us/powershell/

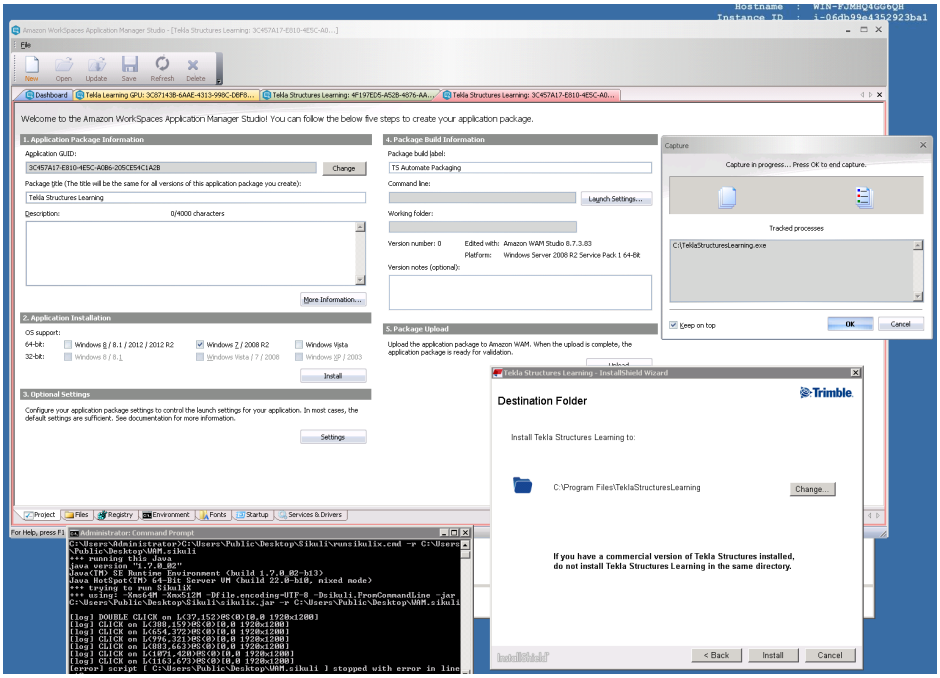


Figure 7.2: A screenshot of EC2 Admin Studio virtualizing Tekla Structures under SikuliX script

Player to validate Tekla Structures. When the validation succeeds, this SikuliX script calls the third Lambda function that will terminate all WAM EC2 instances, notify the deployment process completion and write the logfile to the WorkSpacesAPI table in Amazon DynamoDB. The virtualized Tekla Structures package is now available in Amazon WorkSpaces Console.

The first operation took approximately 30 minutes from start to finish with the T2.Large EC2 instances. Even the whole procedure executes consecutively until it accomplishes, we are still able to follow the procedure and manipulate at any stages from the AWS WorkSpaces Operation Status tool.

7.1.2 Assigning WorkSpaces to client

In the second operation, we plan to launch two WorkSpaces for two users. Firstly, we as administrators have to create new user accounts for these two users in the "Trim.ble.com" domain. As described in the subsection 6.2.5, there are 3 different ways to add new user to the AD. For the evaluation purpose, we first use traditional option which is the Active Directory Users and Computers tool from the Windows Server EC2 instance. We add the first user log-on name together with their first name, last name, email and

password. The second user account is created via **Dsadd** command-line tool built into AWS command line. Both techniques accomplished the new user account creation.

Next, we use Windows PowerShell to send a POST request to the Amazon API Gateway that will trigger to the Lambda function developed for the second operation. We promptly receive the request response that presents the request result and the currently launching WorkSpace's information. The WorkSpace is available within 20 minutes from the time the POST request was sent. The WorkSpace will automatically deliver its access information to the user's email.

7.2 Amazon WorkSpaces Console

Amazon WorkSpaces Console is a web application that provides the control of all WorkSpaces pertinent resources. We will use the Console to assign the Tekla Structures package to the two WorkSpaces recently created. The Console is the only place we can do the assigning task as there is no WorkSpaces API available to do this task at the moment of this evaluation.

There are two sources of applications that can be deployed via Amazon WAM: AWS Marketplace[2] and our own virtualized applications. By default, only AWS Marketplace option is available. In order to use our own application that is Tekla Structures, we have to subscribe for a WAM Standard feature. Once the subscription is activated, we can publish the Tekla Structures package into our CAD and CAM catalog so that it is ready to be assigned to WorkSpace users.

In the **Applications** tab, we can start to assign Tekla Structures to users. We select 2 new users previously created in the *Trim.ble.com* directory to provide access to the Tekla Structures application. In the configure options, we set Installation type as *Required* and Auto update as *Yes*. As a result, Tekla Structures will be forced installed and automatically updated whenever a new version is released. Once these users start their WorkSpaces, Tekla Structures will be installed and available by default. The application also shows up in the Amazon WAM client tool on the users' WorkSpaces. Besides that, there is an *Optional* Installation type that gives end users the capability to decide whether they want to install the software. Additionally, we can monitor the application usage, e.g., assigned and activated applications in the Amazon WorkSpaces Usage tab.

Furthermore, Amazon WorkSpaces Console supports a sharing feature

[2]https://aws.amazon.com/marketplace/cp/WAMProducts

Table 7.1: Hardware configurations comparison

	Amazon Graphics WorkSpaces	Citrix XenDesktop	Tekla Structures Rec.
Operating system	Windows Server 2008 R2	Windows Server 2012 R2	Windows 10 (64-bit)
Processor	Intel® Xeon® E5-2670, 2.60 GHz	Intel® Xeon® E5-2680, 2.50 GHz	Intel® Core i5, 2+ GHz
Memory	15 GB	32 GB	8+ GB
Hard disk	200 GB, SSD	320 GB, SSD	240-480 GB, SSD
Graphics card	NVIDIA GRID K520, 4 GB	NVIDIA GRID K1, 1 GB	NVIDIA GeForce GTX 1060

that allows software owners to share their virtualized packages with other AWS accounts. In other words, organizations that have AWS accounts can directly use the virtualized Tekla Structures package or create their own application package based on the Tekla Structures package that we shared. Additionally, setting up new directories or WorkSpaces can also be done manually in Amazon WorkSpaces Console. Currently, we have to do the management and assigning applications to the users' WorkSpaces manually via Amazon WorkSpaces Console as there are no other ways to handle these tasks.

7.3 Tekla Structures operation in Amazon Graphics WorkSpaces

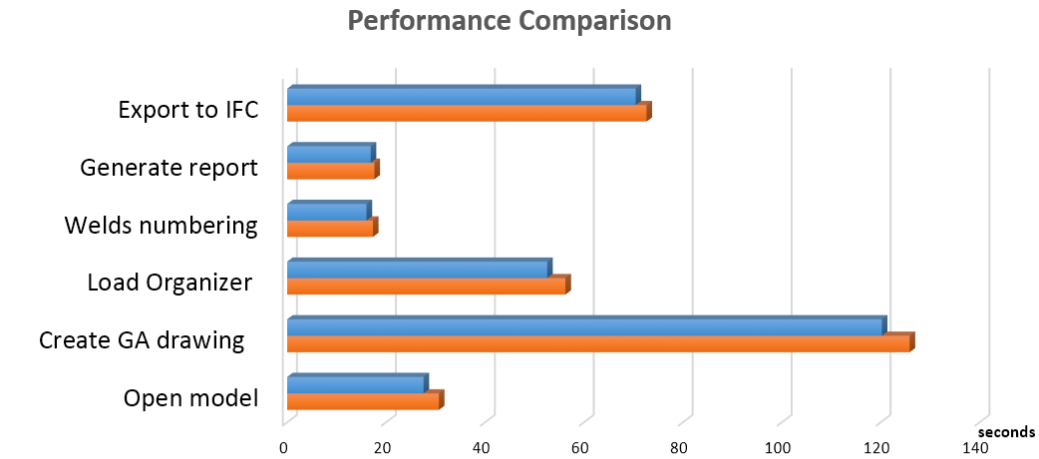
In this section, we execute various assessments with Tekla Structures in Amazon WorkSpaces. The Amazon Graphics WorkSpace that has been created in the previous section is utilized for these evaluations.

7.3.1 Performance evaluation

In order to demonstrate the performance capability of Amazon Graphics WorkSpaces (WorkSpace) with Tekla Structures, we carry out a general operational test. The main objective of this test is to give a reference on how the WorkSpace's performance is compared with another desktop virtualization system. The comparing system that we utilize in this test is the case company's current Citrix XenDesktop virtual machine[3]. The hardware configurations of WorkSpace and the case company's Citrix XenDesktop virtual machine (Citrix) along with Tekla Structures recommendation are shown in the Table 7.1. Since these two systems have dissimilar hardware configurations as well as utilize disparate virtualization technologies, this test is not a definitive comparison to identify the best virtualization system.

The test case aims to measure the time durations WorkSpace and Citrix take to accomplish different tasks in Tekla Structures. The 3D BIM model

[3]https://www.citrix.fi/products/xenapp-xendesktop/



	Open model	Create GA drawing	Load Organizer	Welds numbering	Generate report	Export to IFC
Amazon Graphics WorkSpaces	27.56	120.23	52.57	16.04	16.89	70.45
Citrix XenDesktop	30.67	125.85	56.26	17.33	17.65	72.65

Figure 7.3: Amazon Graphics WorkSpace and Citrix XenDesktop performance comparison chart

used for the test is a relatively large and complex shopping center consisting of approximately 200 000 objects. The reason for choosing such composite model is because we want to employ the system graphics and other resources to their fullest. There are 6 main tasks that we implement in the test: opening the full 3D model, creating a general arrangement drawing, loading the model organizer, numbering model's welds, generating a full report and exporting to IFC file. Windows Performance Analyzer is used to record the time taken by each task. The output is the time to completion of each task in WorkSpace and Citrix as shown in the Figure 7.3.

As can be seen in Figure 7.3, both WorkSpace and Citrix are capable of handling all the Tekla Structures tasks in timely manner. There is no bottleneck in the WorkSpace's resources when processing heavy computational tasks. The time to completion for all the tasks in WorkSpace is slightly faster than in Citrix. It might be explained by the fact that WorkSpace has higher hardware configurations than in Citrix, particularly in GPU computational power. Moreover, Amazon WorkSpaces virtualization technology allows the WorkSpace to have its dedicated virtual hardware for each user while in the Citrix XenDesktop, the Citrix virtual machine has to run several sessions simultaneously for different users.

7.3.2 Interoperability evaluation

In the interoperability evaluation, we deploy Tekla Structures virtualized package in the WorkSpace via Amazon WAM. Besides that, we install the Tekla Structures UK environment, an extension for Tekla Structures, and Trimble Connect[4], a standalone software, with traditional Windows installer packages, i.e., *.msi installers to the WorkSpace.

The result shows that all the applications can be set up as they are in a physical Windows desktop computer. Tekla Structures can recognize and load the installed environment and extension in the first place. Trimble Connect also operates normally without any issues.

Due to the Tekla Structures Learning edition limited functionality and the case company network policies, we could not examine some advanced features that official Tekla Structures provides such as License Administration tool, Tekla multi-user and Tekla model sharing service. Nonetheless, we firmly anticipate that all these features would function properly as if they were on a desktop computer.

7.3.3 Security evaluation

Amazon WorkSpaces is capable of encrypting root and user volumes with Amazon Elastic Block Store. It means that the data stored at rest, disk input/output to the volumes and the volumes' snapshots are thoroughly encrypted. Furthermore, the WorkSpaces' user volume is automatically backed up every 12 hours. In case of a WorkSpace failure, the volume can be simply restored from the backups. Besides that, as a cloud service user of AWS, we are acknowledged of the location of the data center where the WorkSpaces are running and the data is stored.

The PCoIP remote display protocol is utilized to deliver encrypted WorkSpaces' pixels from the data center to be displayed in Amazon WorkSpaces clients. By using the PCoIP protocol, data never actually leaves the data center. Hence, the risks of being intruded and losing important data during the communication is minimized. However, there is a trade-off with this protocol: as no data is sent out, limited data can be received by Amazon WorkSpaces. For instance, Amazon WorkSpaces can only receive input data from touch, mouse and keyboard. Other local peripheral devices such as external hard drives, webcams or local printers will not be able to work with WorkSpaces. To diminish this deficiency, WorkSpaces supports connecting to a network drive and Amazon S3 service as external storage solutions.

[4]http://connect.trimble.com/

7.3.4 Reliability evaluation

The reliability of WorkSpaces when working with Tekla Structures is one of the main concerns for the case company. In order to assess the WorkSpaces' reliability, we work manually with Tekla Structures on a WorkSpace for 6 hours per day for 3 consecutive days as a standard Tekla Structures user would use it. Various tasks are performed in a complex 3D model in Tekla Structures. We also switch occasionally between different Amazon WorkSpace client devices during the usage, namely, Windows desktop, iPad Air and an Android tablet emulator.

As a result, the WorkSpace provides a smooth and stable experience throughout the usage. We encountered no hanging or blackout in the WorkSpace and the clients. Besides that, the input/output latency between Amazon WorkSpace client and the WorkSpace is microscopic. The state of the applications and data are preserved when we disconnect from the WorkSpace. Thus, whenever we resume the WorkSpace after it is stopped, we can get to where we left off, all running applications and data are intact. One more advantage we found during the experiment is that if a WorkSpace faces any fatal errors, it can be completely rebuilt as a brand new machine or from a backup point in a very short time.

7.4 Summary

By the end of this chapter, we have presented the examination of the automation prototype and Tekla Structures operation in the Amazon WorkSpaces.

In general, the prototype functions correctly as we described in the Implementation chapter. The prototype managed to automate successfully all the defined tasks. However, supervision of the Tekla Structures packaging operation in Amazon WAM Admin Studio remains necessary as SikuliX scripts running on these EC2 instances are precarious and susceptible to errors. In other words, a small change in the environments or unexpected procedures occurred while the SikuliX scripts are performing might conduce to the failing of the whole operation. However, the robustness of the automation prototype will be improved when we replace the SikuliX script with the Amazon WAM API when it becomes available.

Furthermore, we are aware that the package we used for the evaluation, i.e., Tekla Structures Learning is a traditional Windows Installer package, not a native virtualized one. Hence, some issues with the software in the virtualization environment may occur, e.g., invalid fonts and missing folders. These issues would not happen if the package was built for virtualization

environment in the first place.

We conclude that Amazon WorkSpaces with Graphics configuration is feasible to use with Tekla Structures. The Amazon Graphics WorkSpaces can handle graphics-intensive tasks with complex 3D model in Tekla Structures. Nevertheless, there are some minor limitations with the Amazon WAM system that needs to be addressed. For instance, administrators have to manually assign the software for specific users or WorkSpaces in Amazon WAM Console. The delay when the software is assigned from Amazon WAM Console to the time WorkSpaces' end users can get the software in Amazon WAM client is relatively long.

Chapter 8

Discussion

This chapter presents the overall benefits and challenges of deploying and utilizing the case company software in Amazon WorkSpaces. We point out some main issues that we have to deal with throughout the implementation. Further developments for a decent automation deployment system are discussed. The new business models of delivering Tekla Structures with the Amazon WorkSpaces are proposed. Moreover, we address alternative virtualization systems and cloud services that could benefit the case company in the future.

8.1 The implementation and future enhancements

The automation prototype developed in this thesis is capable of automating the major processes of packaging a software to Amazon WorkSpaces as well as launching new WorkSpaces to end users. Seven different Amazon Web Services are utilized in the automation prototype and all the services are closely connected to assemble one working system. We built the prototype as an AWS serverless service by combining Amazon API Gateway with AWS Lambda. Hence, there is no administrative effort needed. In order to initiate the prototype, administrator can simply send a valid POST request to invoke the predefined uniform resource locator (URL). Upon receiving the request, the Amazon API Gateway will trigger the automation procedure for virtualizing and validating a software to Amazon WorkSpaces. The similar architect applied for launching WorkSpaces to end users in which a PowerShell script is utilized to add new users' information to an AD and send a different set of POST requests to the Amazon API Gateway, which then forwards the con-

tents to the AWS Lambda to launch new WorkSpaces. We used four separate Lambda functions in the prototype instead of combining them together as one since AWS Lambda is charged according to the requests' compute time not the numbers of available Lambda functions. Hence, dividing into smaller Lambda functions proves to be more cost-efficient and easier to manage.

Even though the prototype fulfills the automation requirements, it is relatively rudimentary and there is room for improvements. The prototype has a basic method to verify POST requests. In order to use for the production, another layer of protection such as applying Client-Side Secure Sockets Layer (SSL) Certificates should be added, so that the system protection is more secured when receiving REST calls from the public network. The prototype log files' information can be adjusted based on the needs and usage in specific scenarios. In further development, we can also integrate the launching WorkSpaces operation in the prototype to the case company customer relationship management system. Thus, we will have a thorough and expeditious automation software delivery system from the time customers purchase the software to the time they get the ready-to-use WorkSpaces with the software installed.

There are few steps in the deployment process that we have to do manually such as assigning the application package to WorkSpaces. Moreover, we have to use the SikuliX script to automate the interaction tasks with Amazon WAM which is error-prone due to the SikuliX technique of detecting matched pixels for automation. Currently, there is no Amazon WAM API or alternative ways to handle these tasks. Hence, this is our solution at the moment. However, replacing the SikuliX scripts and other manual tasks with the Amazon WAM API will be indispensable in the future. Besides that, some function limitations in AWS itself might pose a challenge for the system in the production. For instance, AD Connector in AWS Director Service does not support Security Assertion Markup Language (SAML) or OAuth protocols to communicate with the on-premises AD. Thus, in order to connect with AWS Directory Service the case company AD server will have to be exposed, i.e., directly connected to the AWS environment that may divulge some security risks.

Additionally, the case company software installer, Tekla Structures, is not meant to be built for the virtual environments. Hence, the software requires arduous configurations in Amazon WAM so that the virtualized application container for the software can function correctly in Amazon WorkSpaces. Besides that, as a non-native virtualization software, small changes in operating environment such as missing folders, modified files or incompatible .NET Framework might cause the software to malfunction in WorkSpaces. The optimal solution to solve this issue is to create a native virtual version

of Tekla Structures installer so it will be more consistent with the virtual environment architecture such as Amazon WorkSpaces.

From administrators' perspective, it is essential to be able to monitor the resources of all active Amazon WorkSpaces. We have not contemplated about the issue in this thesis. Nevertheless, Amazon CloudWatch is a suitable AWS service to do those monitoring tasks. In the future development, we can apply Amazon CloudWatch to attain a wider visibility not only into Amazon WorkSpaces' operational health, applications' performance, users' usage but also other Amazon Web Services' resources utilization in the automation prototype. Besides that, applying tags to WorkSpaces is a useful way to identify and organize different WorkSpaces' groups and users.

At the prospect of the development-operations in the case company and the customers' requirements, it will be beneficial to study the integration of the automation prototype to the case company continuous delivery system in TeamCity in order to form a comprehensive continuous deployment system. In essence, any fixes and new developments in the software will be integrated, tested, packaged and deployed to the WorkSpaces expeditiously via the continuous deployment pipeline. Nevertheless, the topic needs to be meticulously researched as the nature of the case company software is a large and composite Windows software, not a simple Web or mobile application.

With the prototype we have built a firm foundation for the software automation deployment in Amazon WorkSpaces. In the future development, based on the specific requirements, detailed infrastructure and resources, we can improve and tailor the prototype to deliver an inclusive solution. For instance, we have to estimate the number of WorkSpaces we want to utilize, their usage periods and whether we will have a dedicated AD or connect the WorkSpaces to the case company AD domain. According to these decisions, we can tweak the automated deployment system to fulfill the discrete production use cases. Furthermore, the AWS serverless service that is a part of the prototype can be reused to process other API requests in further AWS development.

In the Amazon Web Services, we anticipate to have the API support for the Amazon WAM and Amazon WorkSpaces Console in the future. We will be able to replace the SikuliX scripts by the native Amazon WAM API actions in the prototype. Additionally, having Amazon WorkSpaces Console API will allows us to flexibly develop our own assigning and monitoring virtualized applications for Amazon WorkSpaces. Hence, the manual, tedious tasks with the WorkSpaces Console via the web UI can be reduced. Furthermore, we would prefer to have a simpler, more straightforward approach to package the software in preference to using 2 pristine specialized EC2 instances for virtualization and validation each time. With the WorkSpaces, it

will be ideal if AWS can give users the capability to customize the hardware configurations for their WorkSpaces instead of limited to 4 hardware choices currently.

8.2 Delivering the WorkSpace and Tekla Structures as a bundle

Amazon WorkSpaces allows creating a custom bundle for the WorkSpace. In essence, we can create a new WorkSpace, install conventional Windows applications and adjust the WorkSpace settings as required. For instance, we could install any software in the WorkSpace such as Tekla Structures, Google Chrome and Microsoft Office as well as update the system so users can access to the designated network drive. After that, Amazon WorkSpaces Console is utilized to create the bundle that capture this WorkSpace. Consequently, a new bundled WorkSpace launching will have exactly the same installed software and settings as in the bundle. It is a convenient way to deliver the WorkSpaces with a required set of software and Windows preconfigured settings.

However, there is a drawback with this approach. Since the software is not virtualized and installed in a conventional Windows Installer format, Amazon WorkSpaces administrators cannot manage the software in the later stages. Meanwhile with the software packaged with Amazon WAM, administrators are enable to monitor the software, install, uninstall and update the software throughout the whole software life cycle. Nevertheless, there is a possible solution to manage the end users' usage with this approach. Since the WorkSpaces work like traditional Windows desktops and are compatible with popular management tools such as SCCM, we can connect the WorkSpaces to an AD domain and control them with the SCCM system as normal desktops. A future in-depth study for this approach would be beneficial as it could possibility reduce the complexity of virtualizing and managing the software in the Amazon WorkSpaces.

8.3 Business models for software delivery

Tekla Structures uses perpetual licensing model along with maintenance contract to deliver the product for customers. While Amazon WorkSpaces charges monthly or hourly based on the number of running WorkSpaces and the WorkSpaces' hardware configurations. Therefore, by combining the

WorkSpaces with Tekla Structures as a bundle, we can introduce new subscription model in which customers are able to subscribe for a period of time they want to use the bundle.

From the case company perspective, the subscription business model can be designed in order to generate a greater revenue stream from the recurring subscriptions in comparison to the perpetual one-time purchases. Additionally, with the bundle subscriptions, the case company knows the number of active customers and how they use the bundle as a whole and Tekla Structures software in particular as well as the customers churn rate. Furthermore, the bundling subscription model would allow the case company to easily implement cross-selling other products and services. [9, 41]

Enterprise customers can access their business Windows application such as Tekla Structures over the Internet the same way they can use e-mail application in the Web browser. Since there is no upfront investment, the customers only pay for what they use. For a premium product like Tekla Structures, subscriptions allow customers to pay for a period of time. Thus, it could make the product appear to be more affordable. Besides that, in some scenarios where customers only need to use the Tekla Structures for a short period such as for temporary employees or contractors, subscription becomes a big advantage and help to minimize superfluous IT investments as customers do not need to buy a high end workstation and a perpetual Tekla Structures license but to pay for a subscription fee to use the bundle and terminate when it is no longer needed. [12]

Consequently, it is critical to carry out a comprehensive study and field survey to acquire the optimal way for the case company software to license Tekla Structures whether it is a perpetual license, a subscription contract or a hybrid approach involving both methods. Nevertheless, it should be noticed that the availability of Amazon WorkSpaces with Graphics configuration at present is moderately limited. Besides that, the hourly price for the Graphics WorkSpace is still fairly high and there is no monthly price available yet. In the upcoming time, we expect the Amazon WorkSpaces pricing to scale down and the AWS infrastructure to be reinforced.

8.4 Potential cloud services

Desktop Application as a service (DAaaS) is another emerging cloud services that can compete and compliment for Desktop as a Service at the same time. As its name demonstrated, DAaaS delivers the desktop application running in the cloud to end users the same way DaaS offers its cloud desktops. However, DAaaS is more specific in terms and scale as it only streams desktop software

not a whole desktop system as in DaaS. One DAaaS entrant we can try out is Amazon AppStream 2.0[1]. Regardless of its novelty, DAaaS is a highly potential cloud service that we can explore in the future.

Other VDI, cloud-hosted desktops system the case company can contemplate such as VMware Horizon Air and Microsoft Azure RemoteApp. Since each system uses different hypervisors, cloud and virtualization techniques, it will be useful to investigate and test Tekla Structures on these system so that we can have an overall understanding of the advantages and disadvantages each of them possesses.

[1]https://aws.amazon.com/appstream2

Chapter 9

Conclusions

It can be seen that Desktop as a Service is quintessential for today users who require stability, flexibility and mobility in employing full desktop computers. Desktop as a Service appeases users by allowing instant access to high performance Windows desktops at any time, from anywhere and in various device platforms. Furthermore, Desktop as a Service eases the administrator tasks of maintaining, monitoring and upgrading the IT infrastructure as the heavy lifting tasks are handled by the cloud service providers.

In this thesis, we have discussed the concept of the cloud computing and scrutinized the ontology of Desktop as a Service. In addition, the GPU-Accelerated technique in cloud computing was studied. The case company software and their continuous delivery system were investigated. We described and explained the usage of the Amazon Web Services and tools utilized in the thesis implementation.

For the practical part, the thesis has outlined the high level architecture of the whole software deployment process in Amazon WorkSpaces. The working prototype involves 7 different Amazon Web Services and uses Python, PowerShell and SikuliX as development languages. The agile approach of solving the research problem proved to be beneficial as it helps to incrementally develop and enhance the working prototype. In accordance with the deployment process design and usage workflow, we divided the prototype into two operations. The first operation can automatically package the software for the virtualization environment in the Amazon WorkSpaces while the second one is capable of launching new WorkSpaces for end users with minimal administrator's intervention.

The results of the in-depth topical studies about Amazon WorkSpaces and the prototype implementation made it possible to answer the following research questions:

RQ1: How Tekla Structures can be deployed in Amazon WorkSpaces? Will Amazon WorkSpaces be capable of running Tekla Structures?

In order to deploy Tekla Structures in Amazon WorkSpaces, we first have to package the software by virtualizing and validating it for operating accurately in the WorkSpaces. Then Amazon WAM Console is used to manage and assign the packaged software to the WorkSpaces. Besides that, it is possible for end users to install Tekla Structures with the traditional Windows installer package in the WorkSpaces as in normal Windows desktops. However, Amazon WAM Console cannot monitor the usage applications in the WorkSpaces with this installation method. The Amazon Graphics WorkSpaces meet the Tekla Structures hardware recommendations and can run the software smoothly without discernible issues.

RQ2: To what extent can continuous deployment tasks be automated and how to implement these tasks in Amazon Web Services?

In this thesis, we have successfully developed the automation prototype that can handle most of the deployment procedure in Amazon WorkSpaces. There are some minor tasks that need to execute manually due to the lack of supported APIs such as managing and assigning the application to the WorkSpaces. Nonetheless, we expect the APIs to be available in the near future. The prototype was written in Python, PowerShell, SikuliX script and involves several Amazon Web Services that interoperate closely to one another. By using the prototype, administrators can trigger the automation AWS system that we configured to package the software and launch new WorkSpaces without requiring in-depth knowledge of the underlying system.

The major contributions of the thesis work are the proof of concept on how the case company software can be deployed in the Amazon WorkSpaces and the working automation prototype to package the software and launch WorkSpaces for end users in the Amazon WorkSpaces.

Finally, the knowledge cultivated during the thesis work, the automation prototype and the AWS configurations have formed a firm background for the future development of the production system by the case company. We are assured that Desktop as a Service encompasses a lot of potential for the case company to expand their software delivery models to reach a larger and more diverse customer base today and in the future to come.

Bibliography

[1] Agrawal, S., Biswas, R., and Nath, A. Virtual Desktop Infrastructure in Higher Education Institution: Energy Efficiency as an Application of Green Computing. In *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on* (April 2014), pp. 601–605.

[2] Amit, N., Ben-Yehuda, M., Tsafrir, D., and Schuster, A. vIOMMU: Efficient IOMMU Emulation. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2011), USENIXATC'11, USENIX Association, pp. 6–6.

[3] Barr, J. New G2 Instance Type with 4x More GPU Power. `https://aws.amazon.com/blogs/aws/new-g2-instance-type-with-4x-more-gpu-power/`, 2015. (Accessed on 12/29/2016).

[4] Barr, J. New GPU-Powered Amazon Graphics WorkSpaces. `https://aws.amazon.com/blogs/aws/new-gpu-powered-amazon-graphics-workspaces/`, 2016. (Accessed on 12/29/2016).

[5] Bennett, L., and Mahdjoubi, L. Construction Health and Safety, BIM and Cloud Technology: Proper Integration Can Drive Benefits for All Stakeholders. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science* (Dec 2013), vol. 2, pp. 215–218.

[6] Bowker, M., and Matuson, L. Meeting desktop and application delivery challenges across organizations. In *Desktop-as-a-service (DaaS): Greater Operational Control, Reduced Costs, and Secure Workspaces* (Jul 2016).

[7] Casalicchio, E., Iannucci, S., and Silvestri, L. Cloud Desktop Workload: A Characterization Study. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (March 2015), pp. 66–75.

[8] Chen, L. Continuous Delivery: Huge Benefits, but Challenges Too. *IEEE Software 32*, 2 (Mar 2015), 50–54.

[9] Choudhary, V. Comparison of Software Quality Under Perpetual Licensing and Software as a Service. *Journal of Management Information Systems 24*, 2 (2007), 141–165.

[10] Chrobak, P. Implementation of Virtual Desktop Infrastructure in academic laboratories. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on* (Sept 2014), pp. 1139–1146.

[11] Dawoud, W., Takouna, I., and Meinel, C. Infrastructure as a service security: Challenges and solutions. In *Informatics and Systems (INFOS), 2010 The 7th International Conference on* (March 2010), pp. 1–8.

[12] Dubey, A., and Wagle, D. Delivering Software as a Service. *The McKinsey Quarterly 6*, 2007 (2007), 2007.

[13] Grossman, R. L. The Case for Cloud Computing. *IT Professional 11*, 2 (March 2009), 23–27.

[14] Hocke, R. Sikuli / SikuliX Documentation for version 1.1+ (2014 and later) - SikuliX 1.1+ documentation, 2014.

[15] Hou, Q., Qiu, C., Mu, K., Qi, Q., and Lu, Y. A Cloud Gaming System Based on NVIDIA GRID GPU. In *2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science* (Nov 2014), pp. 73–77.

[16] Humble, J., and Farley, D. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series (Fowler)).* Addison-Wesley Professional, 2010.

[17] ITU-T, F. G. Part 1: Introduction to the cloud ecosystem: definitions, taxonomies, use cases and high-level requirements. In *Focus Group on Cloud Computing Technical Report* (Feb 2012).

[18] ITU-T, F. G. Part 2: Functional requirements and reference architecture. In *Focus Group on Cloud Computing Technical Report* (Feb 2012).

[19] ITU-T, F. G. Part 7: Cloud computing benefits from telecommunication and ICT perspectives. In *Focus Group on Cloud Computing Technical Report* (Feb 2012).

[20] Janakiram, M. What's driving the adoption of desktop as a service? `http://www.computerweekly.com/news/2240233145/Whats-driving-the-adoption-of-desktop-as-a-service`, October 2014. (Accessed on 11/07/2016).

[21] Kibe, S., Koyama, T., and Uehara, M. The Evaluations of Desktop as a Service in an Educational Cloud. In *2012 15th International Conference on Network-Based Information Systems* (Sept 2012), pp. 621–626.

[22] Liu, M., Li, T., Jia, N., Currid, A., and Troy, V. Understanding the virtualization "Tax" of scale-out pass-through GPUs in GaaS clouds: An empirical study. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)* (Feb 2015), pp. 259–270.

[23] Luo, S., Lin, Z., Chen, X., Yang, Z., and Chen, J. Virtualization security for cloud computing service. In *Cloud and Service Computing (CSC), 2011 International Conference on* (Dec 2011), pp. 174–179.

[24] Makarov, M., Calyam, P., Sukhov, A., and Samykin, V. Time-based criteria for performance comparison of resource-intensive user tasks in virtual desktops. In *2014 International Conference on Computing, Networking and Communications (ICNC)* (Feb 2014), pp. 112–116.

[25] Martin, R. C. *Agile Software Development, Principles, Patterns, and Practices*. Pearson, 2002.

[26] Mathew, S. Overview of Amazon Web Services, December 2015. (Accessed on 12/07/2016).

[27] Mell, P., and Grance, T. The NIST Definition of Cloud Computing, NIST SP 800-145, Sep 2011.

[28] NVIDIA. GRID Virtual GPU. `http://images.nvidia.com/content/grid/pdf/GRID-vGPU-User-Guide.pdf`, 2016. (Accessed on 12/29/2016).

[29] Orchilles, J. Virtualization: The benefits of VDI. *TechNet Magazine* (2013).

[30] Services, A. W. Amazon Simple Storage Service Developer Guide. `http://docs.aws.amazon.com/AmazonS3/latest/dev/s3-dg.pdf`, 2006. (Accessed on 12/07/2016).

[31] Services, A. W. Aws Directory Service. `http://docs.aws.amazon.com/directoryservice/latest/admin-guide/directory_simple_ad.html`, 2016. (Accessed on 12/29/2016).

[32] Services, A. W. Amazon Elastic Compute Cloud - User Guide for Windows Instances. `http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ec2-wg.pdf`, 2017. (Accessed on 02/03/2017).

[33] Services, A. W. Amazon WorkSpaces FAQs - Virtual Desktops in the Cloud. `https://aws.amazon.com/workspaces/faqs/`, 2017. (Accessed on 02/17/2017).

[34] Silvestri, G. A. *Citrix XenDesktop 5.6 Cookbook*. Packt Publishing, 2013.

[35] Srivastava, A. vDaaS: Reference Architecture. In *2011 Annual IEEE India Conference* (Dec 2011), pp. 1–5.

[36] Suzuki, Y., Kato, S., Yamada, H., and Kono, K. GPUvm: Why Not Virtualizing GPUs at the Hypervisor? In *2014 USENIX Annual Technical Conference (USENIX ATC 14)* (Philadelphia, PA, June 2014), USENIX Association, pp. 109–120.

[37] Trimble. Trimble: Transforming the Way the World Works, 2014. (Accessed on 10/10/2016).

[38] Trimble. Tekla Structures Glossary. `http://teklastructures.support.tekla.com/system/files/files/TeklaStructuresGlossary.pdf`, 2 2016. (Accessed on 10/25/2016).

[39] Waldspurger, C. A. Memory Resource Management in VMware ESX Server. *SIGOPS Operating Systems Review 36*, SI (Dec. 2002), 181–194.

[40] Wang, L., Tao, J., Kunze, M., Castellanos, A. C., Kramer, D., and Karl, W. Scientific Cloud Computing: Early Definition and Experience. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on* (Sept 2008), pp. 825–830.

[41] Zhang, J., and Seidmann, A. Perpetual Versus Subscription Licensing Under Quality Uncertainty and Network Externality Effects. *Journal of Management Information Systems 27*, 1 (2010), 39–68.

[42] Ziglari, H., and Yahya, S. Deployment models: Enhancing security in cloud computing environment. In *2016 22nd Asia-Pacific Conference on Communications (APCC)* (Aug 2016), pp. 204–209.